Designated-Verifier SNARGs with One Group Element

Gal Arnon[1], Jesko Dujmovic[2], and Yuval Ishai[3]

[1]Weizmann Institute and Bar-Ilan University
galarnon42@gmail.com
[2]CISPA Helmholtz Center for Information Security and Saarland University
jesko.dujmovic@cispa.de
[3]Technion and AWS*
yuvali@cs.technion.ac.il

March 19, 2025

Abstract

We revisit the question of minimizing the proof length of designated-verifier succinct non-interactive arguments (dv-SNARGs) in the generic group model. Barta et al. (Crypto 2020) constructed such dv-SNARGs with inverse-polynomial soundness in which the proof consists of only two group elements. For negligible soundness, all previous constructions required a super-constant number of group elements.

We show that one group element suffices for negligible soundness. Concretely, we obtain dv-SNARGs (in fact, dv-SNARKs) with $2^{-\tau}$ soundness where proofs consist of one element of a generic group $\mathbb{G}$ and $O(\tau)$ additional bits. In particular, the proof length in group elements is constant even with $1/|\mathbb{G}|$ soundness error.

In more concrete terms, compared to the best known SNARGs using *bilinear* groups, we get dv-SNARGs with roughly 2x shorter proofs (with 2^{-80} soundness at a 128-bit security level). We are not aware of any practically feasible proof systems that achieve similar succinctness, even fully interactive or heuristic ones.

Our technical approach is based on a novel combination of techniques for trapdoor hash functions and group-based homomorphic secret sharing with linear multi-prover interactive proofs.

*This paper describes work performed at the Technion and is not associated with Amazon.

Contents

1 Introduction

Interactive proofs [GMR89] is a central tool in cryptography and complexity. In an interactive proof system for an NP language L, a prover holding an instance-witness pair (x, w) wants to convince a polynomial-time verifier that $x \in L$, and they do so over multiple rounds of back-and-forth communication. If the statement is true, the verifier should accept; if false, it should accept with probability at most $2^{-\tau}$, no matter what the prover does.

Here, we consider such proof systems with two natural relaxations. First, we only require soundness against computationally bounded provers [BCC88]. Such proof systems are often referred to as *arguments*. Second, we allow the prover and the verifier to engage in an (instance-independent) *preprocessing* protocol. In this setting, we ask the following basic question:

How succinct can a practical proof system be?

By *succinctness*, we refer to the total number of bits exchanged between the prover and the verifier, excluding the preprocessing phase. Originating from the works of Kilian [Kil92] and Micali [Mic94], an enormous body of works studied succinct proof systems that have sublinear communication in the length of the NP-witness.

By *practical*, we loosely refer to proof systems that are practically feasible in the sense that they can run in a "reasonable" amount of time for non-trivial NP-statements, such as proving the satisfiability of a circuit with a few thousand gates. This excludes optimally-succinct proof systems based on general-purpose obfuscation techniques [SW21], which are not yet practical in this sense. To give a more precise notion of practicality, we consider proof systems that can be cast in simple generic models, such as the random oracle model (ROM) [BR93], the generic group model (GGM) [Sho97], or a generic bilinear group model. This is general enough to capture the most succinct proof systems from the literature that are practical in the above informal sense.[1]

Finally, in the above generic models one can typically obtain a *non-interactive* proof system with little to no loss of succinctness. We will therefore restrict our attention to such proof systems, referred to as *succinct non-interactive arguments* (SNARGs) [Mic94, GW11].[2] Here, the preprocessing phase may produce a (possibly long and structured) *common reference string* (CRS), which can be used by any prover. SNARGs where the verifier generates the CRS and may keep a secret verification key are referred to as *designated-verifier* SNARGs (dv-SNARGs).

Concrete succinctness of known practical SNARGs. We briefly summarize the sate of the art on practical succinctness and defer a more detailed overview to Section 1.3. When referring to concrete proof size, we assume 2^{-80} soundness at a 128-bit security level.[3]

- Using generic bilinear groups, there are SNARGs with 2 $\mathbb{G}_1$-elements and 2 field elements [Gro16, Lip24, DMS24]. When instantiated, these yield 1280-bit proofs.
- In the GGM, [BIOW20] obtain a dv-SNARG with 2 $\mathbb{G}$-elements that yields 512-bit proofs, but whose soundness error is inverse-polynomial in the verifier's running time. Obtaining negligible soundness using the [BIOW20] construction requires a super-constant number of $\mathbb{G}$-elements.

This leaves open two questions: Can we obtain negligible soundness in the GGM with a constant number of group elements? Can we obtain *any practical proof system* that meets the above concrete soundness level using fewer than 1280 bits?

[1]The generic models we consider exclude practical *lattice-based* proof systems, e.g., [BISW17, BS23, SSE+24, AFLN24] and many more. However, such proof systems are not competitive with group-based systems in terms of concrete succinctness.

[2]All positive results we refer to also hold for the stronger notion of SNARK [BCC+17]. Moreover, we are not aware of any *interactive* (even heuristic) proof systems that achieve a similar level of succinctness to the non-interactive ones we obtain here.

[3]By 2^{-80} soundness at a 128-bit security level we refer to provable soundness error of 2^{-80} against polynomial-time malicious provers in the standard GGM, where we instantiate the group to have 256-bit elements. A similar convention is used in prior related works, see Appendix A for further discussion.

1.1 Our Results

We answer both questions in the affirmative. We construct the first designated-verifier SNARGs in the generic group model with negligible soundness error and proof size equating to a constant number of group elements. In fact, our proofs consist of a single group element and an additive term that depends on the soundness error. In concrete terms, we can obtain 2^{-80} soundness at a 128-bit security level using 695 bits, almost a 2x improvement over the best pairing-based SNARGs [Gro16, Lip24, DMS24].

Settling for a constant soundness error (say, 1/2), which may be good enough for some practical use cases, the total proof size is close to a *single* group element, almost a 2x improvement over the best previous dv-SNARGs in this setting [BIOW20].

The above results are obtained via two variants of the same blueprint. We begin by describing a SNARG with one group elements and $O(\tau)$ additional bits, and then discuss its construction.

Theorem 1.1 ($1\ \mathbb{G} + O(\tau)$ bit dv-SNARG, informal)**.** *Let $\mathbb{G} = \mathbb{G}_\lambda$ be a generic group*[4] *and τ a soundness parameter. There exists a* dv-SNARG *for proving the satisfiability of a Boolean circuit of size s with the following features:*

- *Soundness error: $2^{-\tau} + O(t^2 \cdot 2^{-\lambda})$ against t-query adversaries;*
- *Proof size: 1 $\mathbb{G}$-element (λ bits) and $O(\tau)$ additional bits;*
- *CRS size: $O(\tau s)$ $\mathbb{G}$-elements.*

See Corollary 7.1 for a formal statement.

In fact, we prove that our dv-SNARG has the stronger notion of knowledge soundness. While the above $O(\tau)$ term in proof size hides a large constant, we show that if we relax the CRS size to $O(\tau s^2)$, the proof can include 1 $\mathbb{G}$-element and 56τ bits. Thus, for use-cases where only a constant soundness error is required, the proof size in the above dv-SNARG is *smaller than* 2 *group elements.* Finally, if we only require *some constant* soundness error $\delta < 1$, then we can get all the way down to 1 $\mathbb{G}$-element and only 7 additional bits.

We note that even a quadratic CRS size may be tolerable when using our dv-SNARK as an "inner system" for proving the correctness of a fast to verify proof generated by an "outer" SNARG, such as Groth16 [Gro16].

Our prover and verifier both make $\tilde{O}(\lambda\tau s)$ group operations (the prover time becomes $\tilde{O}(\lambda\tau s^2)$ when considering the quadratic CRS variant). We leave a more refined optimization of asymptotic and concrete efficiency to future work, and discuss some possible routes for improvement in Section 1.2.

Theorem 1.1 is proved by extending the BCIOP compiler [BCI+13], which combines a "linear-only" encryption scheme and a linear PCP to construct dv-SNARGs, to *compressible* encryption schemes, where ciphertexts can be compressed following homomorphic evaluations. Specifically, we consider the *packed* ElGamal encryption scheme implied by techniques developed in [BGI16, DGI+19, BBD+20]. We analyze the malleability of this encryption scheme and design (variants of) linear PCPs that support instantiating the compiler with the packed ElGamal scheme. In more detail, we show that packed ElGamal in the generic group model is *isolated homomorphic*, which is a form of limited homomorphism which allows the adversary more ability than in the linear-only definition of [BCI+13]. We then use packed ElGamal to construct a dv-SNARG using the compiler along with a *strong linear multi-prover interactive proof* (strong LMIP). While in a standard MIP, the verifier interacts with multiple provers which are unable to share information about the verifier's queries, a strong LMIP additionally requires the honest prover strategy to be linear while retaining soundness against malicious provers with arbitrary strategies. To construct strong LMIPs, we use a linear PCP (either the 1-query LPCP of [BHI+24] for linear CRS size or, for better concrete succinctness with quadratic CRS, the Hadamard-based 2-query LPCP from [BIOW20]) and transform it into a strong LMIP. Such a transformation was first used in [IKO07], and we give an alternate construction (for 2-query LPCPs) which achieves better concrete parameters. See Section 2 for more details.

[4]Here $\mathbb{G}_\lambda$ refers to a generic group of size $\approx 2^\lambda$ and whose elements are described using λ bits.

Improving concrete proof size via hashing. The primary difficulty in constructing the dv-SNARG of Theorem 1.1 is to analyze precisely what power an adversary has in the malleability of the packed ElGamal scheme, and this is the main limitation for achieving smaller proof size. We show that by utilizing a random oracle (which we use only for its collision-resistant properties) it is possible to restrict the adversary's range of actions to a more limited set of malleability attacks. By using this this variant of the packed ElGamal scheme, we design a SNARG whose length is one group element, one output of the random oracle, and a number of bits that tends towards 2τ:

Theorem 1.2 ($1\ \mathbb{G} + 1\ \mathbb{H} + \sim 2\tau$ bit dv-SNARG, informal)**.** *Let $\mathbb{G} = \mathbb{G}_\lambda$ be a generic group, $\mathbb{H} = \mathbb{H}_\lambda$ be a random oracle with λ output bits, τ a soundness parameter, and $p > 2$ be a prime. There exists a* dv-SNARG *for proving the satisfiability of a Boolean circuit of size s with the following features:*

- *Soundness error: $2^{-\tau} + O(t^2 \cdot 2^{-\lambda})$ against t-query adversaries;*
- *Proof size: 1 $\mathbb{G}$-element (λ bits), 1 $\mathbb{H}$ output (λ bits), and $\lceil \frac{2\tau \log p}{\log p - \Theta(1)} \rceil$ additional bits;*
- *CRS size: $O(\tau s \cdot \mathsf{poly}(p))$ $\mathbb{G}$-elements.*

See Corollary 7.2 for a formal statement.

If we allow the CRS size to be quadratic in s, we can make the constant $\Theta(1)$ in the proof size equal to 1. Thus, for soundness error 2^{-80} at a 128-bit security level (i.e., setting $\tau = 80$, $\lambda = 256$) and choosing p to be a 256-bit prime, the proof length is only $2\lambda + \lceil \frac{2\tau \log p}{\log p - 1} \rceil = 695$ bits, almost a 2x improvement over the best pairing-based SNARGs.

As in Theorem 1.1, we rely on linear PCPs (again, using the linear PCPs of [BHI$^+$24] and [BIOW20]). However, due to the reduced malleability of the encryption scheme, we do not have the additional overhead of compiling to strong LMIP. We still need to slightly adapt the PCPs, but this adaptation significantly more efficient. It is due to this that Theorem 1.2 achieves better concrete parameters than Theorem 1.1, at the cost of the added random oracle output. See Section 2.3 for more details.

1.2 Open Problems and Future Directions

In this work, we establish the practical feasibility of dv-SNARGs in the GGM whose proof size contains a single group element and a small number of additional bits that depends on the level of soundness. While this proof size is not too far from optimal, our results leave room for three kinds of improvement: (1) further improving succinctness, (2) improving prover and verifier runtimes, and (3) making the CRS fully reusable. We elaborate on each goal separately below.

Succinctness. There are two plausible approaches for further improving the proof size.

- *Tighter analysis.* In our analysis of packed ElGamal, we give a bound on the possible malleability attacks a malicious party may do. However, we believe that our analysis is quite loose, and conjecture that an even a slightly simpler construction can achieve a better level of soundness. See Section 2.3 for further discussion, as well as an explicit proposal for a dv-SNARG that we conjecture to achieve soundness $2^{-\tau}$ with proofs consisting of only 1 group element and $\approx 2\tau$ additional bits. For a soundness error of 2^{-80} at a 128-bit security level, this would amount to a proof size of ≈ 420 bits.
- *Better PCPs.* As with prior constructions [BCI$^+$13, BIOW20, BHI$^+$24], our dv-SNARGs rely on different flavors of linear PCPs. However, unlike these previous constructions, our apporach is less sensitive to the number of queries and depends mainly on the ratio μ between the *total bit-length* of the LPCP answers and the soundness level τ. The LPCPs we use have $\mu \approx 2$, which is why the SNARG described in Theorem 1.2 tends to 2τ, and explains the 2τ additive term in our proof length. As noted in [BHI$^+$24], known hardness of approximation results for MAXLIN [Hås01, FJ12, ABCH19] imply 1-query LPCPs with $\mu \approx 1$. Using such a linear PCP would result in a proof where the additive term is

improved from $\approx 2\tau$ to $\approx \tau$. However, these LPCPs have a non-negligible completeness error and seem practically infeasible. The completeness error can potentially be eliminated by allowing more queries, combining PCPs with optimal amortized query complexity [HK05] with the universal factor graph technique from [ABCH19] to make the query distribution input-independent. However, this approach too seems practically infeasible. We leave open the question of designing practical LPCPs with $\mu \approx 1$.

The above two potential improvements could lead to the following dv-SNARG.

Conjecture 1.3. Let $\mathbb{G} = \mathbb{G}_\lambda$ be a generic group and τ a soundness parameter. There exists a dv-SNARG for proving the satisfiability of a Boolean circuit of size s with the following features:

- Soundness error: $2^{-\tau+\log\log\lambda} + O(t^2/2^\lambda)$ against t-query adversaries;
- Proof size: 1 $\mathbb{G}$-element and $\tau + o(\tau)$ additional bits;
- CRS size: $O(\tau s)$ $\mathbb{G}$-elements.

For a soundness error of 2^{-80} at a 128-bit security level this would amount to a proof size of ≈ 340 bits, roughly half the proof size from Theorem 1.2.

Runtimes. Our new proof systems are practically feasible even for satisfiability problems involving thousands of constraints. However, the concrete runtime of the prover and (especially) the verifier still leave much to be desired, and improving these overheads is a major direction for future research.

Our packed ElGamal encryption is based on the distributed discrete logarithm algorithm from [BGI17], which helps us achieve perfect completeness. With a more careful analysis, one might be able to switch to the faster distributed discrete logarithm algorithm from [DKK18] to quadratically reduce verification time.

An orthogonal improvement is a tighter analysis of the SNARG verification time. Our analysis pessimistically assumes the magnitude of LPCP answers to scale linearly with the proof length. However, for natural linear PCPs, a quadratic improvement could be potentially obtained by using a concentration bound for a corresponding random walk. Similar ideas have been explored in [BIOW20]. In Claims 6.5 and 6.10, we show that this analysis is compatible with our transformations. See Conjecture 3.11 for a relevant conjecture.

Combining both of the above potential optimizations, the verifier's runtime can grow linearly with $s^{1/4}$ rather than linearly with s, potentially making our SNARGs practical for much larger circuits.

Finally, there is a lot of room for improving the concrete efficiency of the LPCPs we employ. In particularly, we rely on 1-query LPCPs from [BIOW20, BHI$^+$24] that apply to Boolean constraints or arithmetic constraints over small fields. Extending them to natively accommodate arithmetic constraints over large fields remains open.

Reusability. In every dv-SNARG, the CRS setup can be safely reused an arbitrary number of times as long the prover does not learn (too many times) whether the verifier accepts badly formed proofs.[5] This may be good enough for many practical use cases, especially when there are long-term relations between the prover and the verifier. In particular, the verifier can replace the CRS after several rejections, or alternatively not reveal whether each individual proof is accepted.

However, the standard (strong) notion of reusability for SNARGs requires that the CRS can be safely reused even when a malicious prover can fully observe the verifier's accept/reject decisions. Our dv-SNARGs are not reusable in this sense, leaving the question of achieving full reusability open. We explain the source of the problem below.

The BCIOP compiler [BCI$^+$13] shows that by combining an LPCP that has reusable soundness with a linear-only encryption scheme, one can obtain a dv-SNARG with (strong) reusable soundness. However, the kinds of LPCPs and MIPs we use (concretely, "strong linear MIPs" and "modded linear PCPs") do not have reusable soundness. In the case of strong linear MIPs, where malicious provers can employ an arbitrary

[5]In contrast, some *interactive* arguments in the preprocessing model, such as ones suggested in [IKO07, BHI$^+$24], require an independent setup for each proof instance even when malicious provers cannot learn the verifier's decisions.

strategy, the lack of reusable soundness seems inherent. However, there is hope to construct reusably sound modded LPCPs. Indeed, [BHI$^+$24, Corollary 5.24] realized reusably sound *bounded* 1-query LPCP over large fields, which is a strongly related notion.

1.3 Related Work

In this section, we give an overview of the concrete level of succinctness that can be achieved in each of the main generic models: the random oracle model (ROM), the generic bilinear group model, and the generic group model (GGM). For concreteness, we require here soundness error of 2^{-80} at a 128-bit security level for designated verifier SNARGs and 128-bit soundness for publicly verifiable ones. See Appendix A for an extended discussion about the security notion and this choice of numbers for comparison.

Random Oracle Model. In the ROM, the most succinct hash-based SNARGs [BBHR18, ACFY24a, ACFY24b] have proofs with size roughly 4Kib (for instances of size 2^{12}), At the technical level, these (setup-free) SNARGs combine the blueprint of Kilian and Micali with an interactive variant of classical PCPs known as an IOP [BCS16, RRR16]. See [CY24] for further details. Using classical PCPs instead of IOPs, one could potentially obtain somewhat better succinctness at the expense of a much slower prover time. However, even in this case, proofs would have length in the thousands of bits.

Generic Bilinear Group Model. Bilinear group-based SNARGs can obtain a much better level of succinctness by incorporating a different relaxation of classical PCPs known as a *linear PCP* [IKO07]. The first practical SNARGs based on bilinear groups were given by Groth [Gro10]. Following a sequence of works [Lip13, GGPR13, BCI$^+$13, DFGK14], the Groth16 SNARG [Gro16] was considered until recently to be the state of the art in succinctness. Built on asymmetric pairings, a Groth16 proof has size 2 $\mathbb{G}_1$-elements and 1 $\mathbb{G}_2$-element. For the popular group of choice, BLS12-381 (for 128-bit security), this corresponds to $2 \cdot 384 + 768 = 1536$ bits. Recently, [Lip24] improved on the size of Groth16, achieving a size of 3 $\mathbb{G}_1$-elements and 1 field element, which equates to $3 \cdot 384 + 256 = 1408$ bits. This was reduced in [DMS24] to 2 $\mathbb{G}_1$-elements and 2 field elements arriving at $2 \cdot 384 + 2 \cdot 256 = 1280$ bits. When instantiating [Mic94] with the linear map commitments of [LM19] and a 2-query linear Reed-Solomon PCP implicit in [DFGK14] (see [BHI$^+$24, Corollary D.6]), one also gets a proof size of 2 $\mathbb{G}_1$-elements and 2 field elements. If one is willing to use the full PCP machinery instantiating [Mic94] with the subvector commitments of [LM19], one may get a slightly lower proof size. However, as discussed above, such general-purpose PCPs have poor concrete efficiency.

Generic Group Model. Most relevant to our work, another line of research [BCI$^+$13, BCC$^+$16, BBB$^+$18, BIOW20, BHI$^+$24] considers minimizing proof size 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. Settling for inverse-polynomial soundness error, Barta et al. [BIOW20], obtained dv-SNARGs with proofs as short as 2 $\mathbb{G}$-elements, which corresponds to 512 bits for Curve 25519, a popular group of choice. However, applying this construction with our soundness target of 2^{-80} would make verification practically infeasible, unless the proof size is increased drastically to amplify soundness.

1.4 Organization

The rest of this paper is organized as follows. In Section 2, we give a high-level overview of the ideas and techniques used in our work. In Section 3, we introduce preliminary notation, definitions and results known from prior work. In Section 4, we define compressible encryption schemes, and introduce the packed ElGamal encryption scheme, along with a variant thereof. In Section 5, we define malleability security notions and prove that our compressible encryption schemes meet these security guarantees. In Section 6, we show how to transform linear PCPs to strong linear MIPs and to modded linear PCPs. In Section 7, we combine compressible encryption schemes and suitable linear PCPs (or MIPs) to construct dv-SNARGs.

2 Technical overview

In this section, we give an overview of our results and the underlying techniques.

2.1 Designated-verifier SNARGs from compressible encryption

We revisit a paradigm for constructing designated-verifier SNARGs by combining linearly homomorphic encryption schemes with linear PCPs and related objects (such as linear IPs) developed in [IKO07, BCI⁺13]. In a linear PCP (LPCP) over a field $\mathbb{F}_p$ the prover (whether honest or malicious) commits to a proof $\pi \in \mathbb{F}_p^\ell$, and the verifier chooses queries $\mathbf{a}_1, \ldots, \mathbf{a}_q \in \mathbb{F}_p^\ell$. The verifier then receives answers $(b_1, \ldots, b_q) \in \mathbb{F}_p$ to the queries, where $b_i = \langle \pi, \mathbf{a}_i \rangle \in \mathbb{F}_p$. To construct a designated-verifier SNARG from LPCPs, we have the verifier choose its queries first and put them into the common reference string. To have any chance of preserving soundness, we hide these queries from the prover by encrypting them, where only the verifier is given the decryption key. Intuitively, this way the prover has a hard time making its proof string depend on the queries. For completeness to still hold, we need this encryption to enable linear computation on the encrypted messages.

We describe in more detail the transformation given an LPCP and an encryption scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ that is linearly homomorphic. For convenience of notation, for now, we consider only 1-query LPCPs.

- **Setup.** Generate keys $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}$ for the encryption scheme, and PCP verifier query $\mathbf{a} \in \mathbb{F}_p^\ell$. Encrypt the queries, $\mathsf{ct}_i = \mathsf{Enc}(\mathsf{pk}, \mathbf{a}[i])$, where $\mathbf{a}[i]$ is the i-th entry of $\mathbf{a}$. The verifier private state is sk, and the public reference string contains the public key pk and ciphertexts $(\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$.

- **Prover.** Given pk, and $\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell$, the honest prover generates $\pi \in \mathbb{F}_p^\ell$ as in the linear PCP. It then homomorphically evaluates $\langle \pi, \mathbf{a} \rangle$ using the ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell$, thus generating a new ciphertext ct'. The prover message is ct'.

- **Verifier.** Given the secret key sk, and ciphertext ct', decrypt $b = \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}') \in \mathbb{F}_p$ and check that the PCP verifier accepts given b as the query answer.

Observe that in order to argue soundness, we need more properties from our encryption scheme. Indeed, if a malicious prover can do non-linear operations on the encrypted messages, then it can essentially launch a non-linear attack on the linear PCP, in which case we cannot rely on soundness of the PCP. Thus, we want the encryption scheme to be linearly homomorphic but the homomorphic operations to be restricted only to linear ones (or, more generally, affine ones). Encryption schemes with this property are referred to as being "linear-only" homomorphic.

Designated verifier SNARGs from ElGamal. Following the [BCI⁺13] paradigm, Barta et al. [BIOW20] construct dv-SNARGs by utilizing the ElGamal encryption scheme, which is linearly homomorphic for small messages. Recall that, given a suitable group $\mathbb{G}$ of order p' with generator g, the ElGamal encryption scheme is:

- KeyGen: Sample random $x \leftarrow_\$ \mathbb{Z}_{p'}$. Set $\mathsf{pk} = g^x$ and $\mathsf{sk} = x$.
- Enc: Given public key $\mathsf{pk} = h$, to encrypt $m \in \mathbb{Z}_{p'}$ pick $r \leftarrow_\$ \mathbb{Z}_{p'}$ and output $(g^r, h^r g^m)$.
- Dec: Given secret key $\mathsf{sk} = x$ and a ciphertext (c_1, c_2), compute the message $m = \mathsf{DLog}(c_2 \cdot c_1^{-x})$.

Decryption, here, only works if m is small (i.e., so that discrete log can be computed by polynomially bounded honest parties). [BIOW20] show that, in the generic group model (GGM), the ElGamal encryption scheme satisfies linear targeted malleability, a variant of linear-only encryption. They further design 1-query LPCPs over a field of size $\mathsf{poly}(\lambda)$ with soundness error $1/\mathsf{poly}(\lambda)$. By combining these two ingredients using the compiler described above, they construct a dv-SNARG in the GGM whose argument consists of 2 group elements and whose soundness error is $1/\mathsf{poly}(\lambda)$.

In the following, we explore how to push this idea to negligible soundness.

Adapting [BIOW20] for negligible soundness. The easiest way to achieve negligible soundness using the previous approach is to repeat the proof q times, where q is super-constant in λ. This, however, would increase size of the proof to $2q = \omega(1)$ group elements, which is too large.

Our first step to reduce the size is to reuse the ciphertext randomness of ElGamal with multiple secret keys in order to encrypt a vector of messages $m_1, \ldots m_q$. In more detail:

- KeyGen: Sample random $x_1, \ldots, x_q \leftarrow_\$ \mathbb{Z}_{p'}$. Set $\mathsf{pk} = (g^{x_1}, \ldots, g^{x_q})$ and $\mathsf{sk} = (x_1, \ldots, x_q)$.
- Enc: Given public key $\mathsf{pk} = (h_1, \ldots, h_q)$, to encrypt $m_1, \ldots, m_q \in \mathbb{Z}_{p'}$ pick $r \leftarrow_\$ \mathbb{Z}_{p'}^n$ and output $(g^r, h_1^r g^{m_1}, \ldots, h_q^r g^{m_q})$.
- Dec: Given secret key $\mathsf{sk} = (x_1, \ldots, x_q)$ and a ciphertext $(c_0, \ldots, c_q)$, output $(\mathsf{DLog}(c_1 \cdot c_0^{-x_1}), \ldots, \mathsf{DLog}(c_q \cdot c_0^{-x_q}))$.

Observe that this change preserves linear homomorphism: given the two ciphertexts each encrypting a q-message vector $(g^r, h_1^r g^{m_1}, \ldots, h_n^r g^{m_q})$ and $(g^{r'}, h_1^{r'} g^{m'_1}, \ldots, h_q^{r'} g^{m'_q})$ we can compute a ciphertext

$$\begin{aligned}&(g^r g^{r'}, h_1^r g^{m_1} h_1^{r'} g^{m'_1}, \ldots, h_t^r g^{m_q} h_n^{r'} g^{m'_q})\\ =&(g^{r+r'}, h_1^{r+r'} g^{m_1+m'_1}, \ldots, h_q^{r+r'} g^{m_q+m'_q}),\end{aligned}$$

which decrypts to the sum of the message vectors. Thus, we can use it in the paradigm.

With this modification, we have already reduced our proof length from $2q$ to $q+1$ group elements, which is a significant decrease but still requires a super-constant number of group elements to achieve negligible soundness error. However, we have gained more power: the malicious prover is restricted to computing the same linear function over all elements of the vector. This allows us to move from a 1-query LPCP to a q-query one rather than repeat the 1-query LPCP q times. Multi-query LPCPs are significantly easier to design than their 1-query variant.[6] Revisiting our dv-SNARG construction, we use a q-query LPCP to generate q queries $\mathbf{a}_1, \ldots, \mathbf{a}_q$. The verifier then encrypts the queries $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathbf{a}_1[i], \ldots, \mathbf{a}_q[i])$. As in the 1-query compiler described above, the common reference string contains all of these ciphertexts. The prover then homomorphically computes a ciphertext encrypting the value $\pi^\intercal \begin{pmatrix}\mathbf{a}_1 & \ldots & \mathbf{a}_q\end{pmatrix}$ and sends it to the verifier. The verifier decrypts this ciphertext and checks whether the LPCP verifier accepts given the decrypted values.

We have established that the paradigm can be made to work relatively efficiently for q-query linear PCPs. However, this does not suffice to achieve negligible soundness with a constant number of group elements, as we would need a linear PCP with constant query complexity and negligible soundness, which we only know how to construct over large fields, which is both incompatible with computing the discrete log, and would require a much larger generic group.

Smaller proofs using compressible encryption. We make two observations. The first is that because the messages need to be small to be able to compute the discrete logarithm, most of the group is unused. In other words, in an amortized sense, one group element of size $O(\lambda)$ only encodes $\mathrm{polylog}(\lambda)$ bits of information. Our second observation is that the homomorphic properties of the encryption scheme are used only once in the dv-SNARG. Indeed, after computing the query answers under the encryption, the prover simply sends the resultant ciphertexts to the verifier, who decrypts them immediately.

In order to utilize the above observations, we consider encryption schemes which are *compressible*. In a (linearly homomorphic) compressible encryption scheme, the encryption procedure Enc produces ciphertexts ct that are large and support homomorphism. The scheme additionally has a compression algorithm $\mathsf{Compress}$ takes a ciphertext ct and compresses it into a smaller ciphertext cct, which might lose the homomorphic capabilities held by ct.

We can now restate the transformation using the combined ideas of q-query LPCPs and compressible encryption schemes:

[6]See [BIOW20, BHI+24] for further discussion on the complications in designing 1-query LPCPs.

- **Setup.** Generate keys $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}$ for the encryption scheme, and PCP verifier queries $\mathbf{a}_1, \ldots, \mathbf{a}_q$. Encrypt the queries,
$$\mathsf{ct}_i = \mathsf{Enc}(\mathsf{pk}, \mathbf{a}_1[i], \ldots, \mathbf{a}_q[i]).$$
The verifier private state is sk, and the public reference string contains the public key pk and ciphertexts $(\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$.

- **Prover.**
  1. Given pk, and $\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell$, the prover generates π as in the linear PCP. It then homomorphically evaluates $\langle \pi, \mathbf{a}_1 \rangle$ up to $\langle \pi, \mathbf{a}_q \rangle$ using the ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell$, thus generating a new ciphertext ct'.
  2. Compute a compressed ciphertext cct from ct' using $\mathsf{Compress}$. The prover message is cct.

- **Verifier.** Given the secret key sk, and compressed ciphertext cct, decrypt $(b_1, \ldots, b_q) = \mathsf{Dec}(\mathsf{sk}, \mathsf{cct})$ and check that the PCP verifier accepts given $b_1, \ldots, b_q$ as the query answers.

In the next sections, we explore instantiating this extension of the [BCI+13] paradigm.

2.2 Packed ElGamal

We consider the *packed* ElGamal scheme implied by techniques developed in [BGI17, DGI+19] and coined in [BBD+20]. The setup and encryption of the scheme are identical to multi-message ElGamal, but now we also consider a *compression* algorithm and subsequent decryption algorithm for compressed ciphertexts.

The main ingredient of the compression procedure is the "distributed discrete logarithm" (DDL) algorithm. The DDL algorithm allows two parties that have group elements h_1 and $h_2 = h_1 \cdot g^x$ (respectively), to convert these elements into integers y_1 and y_2 such that $y_1 = y_2 + x$, given that x is smaller than some fixed bound B. We describe a simple distributed discrete logarithm algorithm (more efficient algorithms exist but are unnecessary to understanding our results). Suppose the two parties are given access to a random function $\phi \colon \mathbb{G} \to \{0,1\}^\ell$. Now, each party computes its DDL share as follows: compute $h_i \cdot g^y$ for every $y < B'$, and output the smallest y such that $\phi(h_i g^y) = 0^\ell$. Now, if B' is much bigger than B with respect to x then with high probability both parties will arrive at the same element which maps to 0^ℓ, i.e., $h_1 g^{y_1} = h_2 g^{y_2} = h_1 g^{x+y_2}$, and so $y_1 = y_2 + x$.

We show how to use DDL to compress an ElGamal ciphertext that encrypts a message $m_1, \ldots, m_q \in \mathbb{Z}_p$:

- $\mathsf{Compress}$: Given a ciphertext $\mathsf{ct} = (c_0, c_1, \ldots, c_q)$, compute $v_i = \mathsf{DDL}(c_i) \mod p$ for all $i \in [q]$, and output $\mathsf{cct} = (c_0, v_1, \ldots, v_q)$.
- Dec: Given secret key $\mathsf{sk} = (x_1, \ldots, x_q)$ and a compressed ciphertext $(c, v_1, \ldots, v_q)$, compute
$$\left((\mathsf{DDL}(c^{x_1}) - v_1) \mod p,\ \ldots,\ (\mathsf{DDL}(c^{x_q}) - v_q) \mod p\right).$$

Then, assuming the distributed discrete logarithm algorithm does not fail, computing the compression and then the decryption procedures becomes,
$$\mathsf{Dec}(\mathsf{Compress}(g^r, h_1^r g^{m_1}, ..., h_q^r g^{m_q})),$$
and for every $i \in [q]$ we get:
$$\begin{aligned} &\mathsf{DDL}(g^{x_i r}) \mod p - \mathsf{DDL}(g^{x_i r + m_i}) \mod p \\ =&(y_i - (y_i - m_i)) \mod p = m_i \mod p \end{aligned}$$

Failures of the DDL algorithm can be prevented by the compressing party resampling the randomness of the ciphertext. While the compressor cannot check whether a compression error has occurred since it does not the decryption key, it can test whether there is a possible value which leads to a failure (recall that we are considering here p which is small).

Is packed ElGamal linear only? Recall that in order to use the paradigm to construct dv-SNARGs we needed an encryption scheme that is linear-only (or linear targeted malleable), i.e., is capable of doing linear operations on the encrypted messages and nothing else. Since $\mathsf{Compress}$ is a postprocessing procedure to the standard ElGamal ciphertexts, one could naively expect that this encryption, too, is linear only. However, we show that this is, in fact, not the case by demonstrating that one can homomorphically evaluate non-linear functions by forcing a decryption error.

We demonstrate how to evaluate a non-linear function over a packed ElGamal encryption with a ciphertext that encrypts a message $m \in \{0, 1, 2\}$, and the compression happens modulo 3. The adversary gets a ciphertext $\mathsf{ct} = (c_0, c_1)$ and the public key (g, h). Further, it knows $h = g^x$, $c_0 = g^r$, and $c_1 = g^{rx+m}$ for some $r, x \in \mathbb{Z}_{p'}$ and $m \in \{0, 1, 2\}$. In order to attack the scheme, the adversary scales (c_0, c_1) by a large random number $s \in \mathbb{Z}_{p'}$. More specifically, it produces a new ciphertext $(c'_0 = c_0^s = g^{rs}, c'_1 = c_1^s = g^{rxs+sm})$. If the adversary chooses the (malformed) compressed ciphertext (c'_0, e) for $e \in \mathbb{Z}_3$, then the decryption procedure Dec will output $(\mathsf{DDL}(c_0'^x) - e) \mod 3$. Of course, the evaluator does not know $c_0'^x$, but it does know $c'_1 = c_0'^x / g^m$. Therefore, it knows that Dec will output the following:

- $(\mathsf{DDL}(c'_1) - e) \mod 3$ if $m = 0$;
- $(\mathsf{DDL}(c'_1/g^s) - e) \mod 3$ if $m = 1$;
- $(\mathsf{DDL}(c'_1/g^{2s}) - e) \mod 3$ if $m = 2$.

In other words, the adversary can homomorphically evaluate the function $f_{s,e}$ defined as:

$$m \mapsto \begin{cases} (\mathsf{DDL}(c'_1) - e) \mod 3 \text{ if } m = 0 \\ (\mathsf{DDL}(c'_1/g^s) - e) \mod 3 \text{ if } m = 1 \\ (\mathsf{DDL}(c'_1/g^{2s}) - e) \mod 3 \text{ if } m = 2 \end{cases}$$

Because c'_1, c'_1/g^s, and c'_1/g^{2s} are far apart, the value output of DDL given each as input is independent (as they choose a different zero point of ϕ), and random[7]. Therefore, for example, with a constant probability we will have the function $(\mathsf{DDL}(c'_1) - e) \mod 3 = (\mathsf{DDL}(c'_1/g^s) - e) \mod 3 = 0$ and $(\mathsf{DDL}(c'_1/g^{2s}) - e) \mod 3 = 1$, which is not a linear function over $\mathbb{Z}_3$.

Because s and e are chosen by the adversary, and it can compute $f_{s,e}$, it can also use rejection sampling until $f_{s,e}$ is whatever function it wants.

This attack generalizes to bigger message spaces and multiple ciphertexts. It additionally generalizes in the following sense to ciphertexts encrypting multiple messages: given a ciphertext encrypting a vector $(m_1, \ldots, m_n)$ the evaluator can evaluate non-linear functions $f_1, \ldots, f_n$ on the ciphertext, such it decrypts to $f_1(m_1), \ldots, f_n(m_n)$.

Isolated homomorphism of packed ElGamal. We prove in the generic group model that the above-mentioned attack is the most a malicious party can do. More specifically, we prove that for every adversary provided with ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell$ encrypting vectors $\mathbf{m}_1, \ldots, \mathbf{m}_\ell \in \mathbb{F}_p^q$ that outputs a compressed ciphertext cct, there are functions $f_1, \ldots, f_q$ so that

$$\mathsf{Dec}(\mathsf{cct}) = (f_1(\mathbf{m}_1[1], \ldots, \mathbf{m}_1[\ell]), \ldots, f_q(\mathbf{m}_q[1], \ldots, \mathbf{m}_q[\ell])) \quad .$$

In other words, the adversary can evaluate arbitrary functions, but it cannot share information between different "slots" of the vectors. We call this property of the encryption scheme *isolated homomorphism.*

Strong linear MIPs. Isolated homomorphism of the packed ElGamal encryption scheme allows a malicious prover to apply an arbitrary function f_i to the veifier's i-th query $\mathbf{a}_i$. However, since these functions are isolated, they cannot "share" information about between different slots of the ciphertexts. Thus, isolated homomorphism translates to the soundness of a multi-prover interactive proof (MIP) rather than of a PCP. In an MIP, a set of provers $P_1, \ldots, P_q$ wants to convince a single verifier of a statement. The verifier

[7]It is not uniformly random but the distribution has high enough entropy to make this attack work.

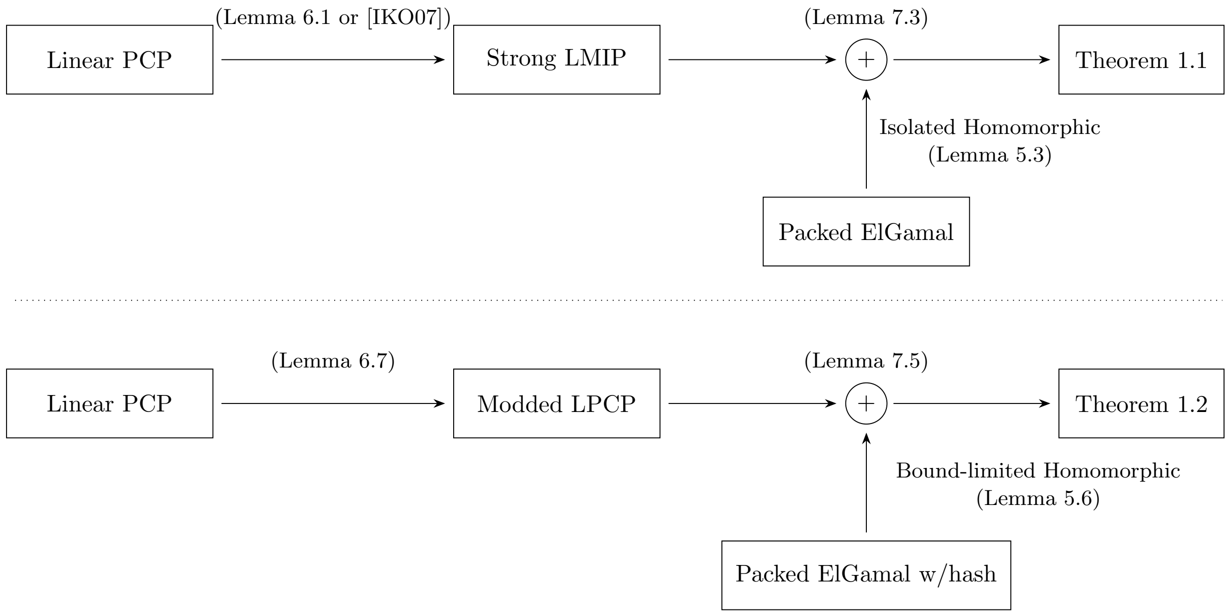

Figure 1: Summary of our transformations.

sends a query $\mathbf{a}_i$ to each prover P_i and receives a response b_i. A malicious set of provers can answer with $(f_1(\mathbf{a}_1), \ldots, f_q(\mathbf{a}_q))$ for any arbitrary functions $f_1, \ldots, f_q$. This exactly matches the guarantees provided by the isolated homomorphism notion.

In fact, we need the additional property that the honest proof to be a single linear strategy. We call an MIP where the honest prover computes a single linear function, but the malicious provers are allowed to compute arbitrary (isolated) functions a *strong linear MIP*[8]. Strong linear MIPs have been constructed (e.g., in [IKO07]) by starting with a linear PCP with soundness against provers that (1) only apply linear strategies and (2) apply the same strategy to each query. These requirements are relaxed by (1) adding a linearity test, thereby removing the linearity assumption, and (2) adding a consistency check between the queries, thereby removing the single strategy requirement.

This construction suffices to get strong linear MIPs with constant soundness, which can then be boosted by repeating the protocol. Unfortunately, the concrete parameters achieved by this process leave much to be desired. We give an alternate transformation that is specific to 2-query linera PCPs which combines the linearity and consistency checks into one combined check inspired by the linear-consistent test of [AHRS01], thus improving the constants derived by this transformation. See Section 6.1 for further technical details.

Dv-SNARGs from packed ElGamal. We combine the packed ElGamal encryption scheme with strong linear MIPs to construct dv-SNARGs. Recall that we had compressed ciphertexts of the form $\mathsf{cct} = (c_0, v_1, \ldots, v_q)$, such that c_0 is a group element, and $v_i \in \mathbb{Z}_p$, where $\mathbb{Z}_p$ is the message space. Thus, the dv-SNARG has proof length equal to one compressed ciphertext of a message of length equal to the size of the query answers in the MIP.

Thus, when instantiated with a linear MIP (small) field $\mathbb{F}_p$, with $O(\tau)$ queries and soundness error $2^{-\tau}$ (which can be constructed from ones with constant soundness via τ-wise repetition), we get a dv-SNARG whose proof length is a single group element along with $O(\tau \cdot \log p)$ bits (Theorem 1.1). Figure 1 provides a summary of our transformations from linear PCP to dv-SNARG.

[8]In [IKO07] this notion is called linear MIP. In more recent work [BISW18], linear MIP refers to a notion in which the malicious prover also has to behave linearly.

2.3 Improved proof length by reducing malleability

The main issues described in the previous section are caused due to the malicious prover having the ability to make DDL fail without being detected. We show that this behavior of the malicious verifier can be limited at the cost of appending a hash of the points that DDL synchronizes to (i.e., the locations where ϕ is zero which the DDL algorithm outputs). Let H be a hash function, modeled by a random oracle. We change the compression and decryption schemes in the following way:

- $\mathsf{Compress}(\mathsf{ct} = (c_0, c_1, \dots, c_q))$:
  1. Let $v_i \leftarrow \mathsf{DDL}(c_i)$ for $i \in [q]$.
  2. Let $k \leftarrow \mathsf{H}(c_1 g^{v_1}, \dots, c_n g^{v_q})$.
  3. Output $(c_0, v_1 \mod p, \dots, v_q \mod p, k)$.
- $\mathsf{Dec}(\mathsf{sk} = (x_1, \dots, x_q), \mathsf{cct} = (c, e_1, \dots, e_q, k))$:
  1. Let $v'_i \leftarrow \mathsf{DDL}(c^{x_i})$ for $i \in [q]$.
  2. If $\mathsf{H}(c^{x_1} g^{v'_1}, \dots, c^{x_n} g^{v'_q}) \neq k$ output $\perp$.
  3. Otherwise, output $((v'_1 - e_1) \mod p, \dots, (v'_n - e_q) \mod p)$

Observe that compressed ciphertexts have size 1 group element, one hash output, and $q \log p$ bits, as opposed to 1 group element and $q \log p$ bits, which seems worse than our previous scheme.

However, we show that the protocol above is bound-limited homomorphic, a notion which enables the prover only slight non-linear power (see Section 5.1 for a formal definition). Importantly, a bound-limited homomorphism is significantly more restrictive than isolated homomorphism. Thus, we do not have to divert to strong linear PCPs, and can model these attacks as a slightly modified version of linear PCPs (which we call modded linear PCPs). We show that standard linear PCPs can be adapted to this modified model with small loss in the query complexity. Known linear PCPs can have significantly better parameters than strong linear MIPs, translating to a smaller number of queries q.

This allows us to construct a dv-SNARG with proof size 1 group element, one hash output, and bits approaching 2τ (Theorem 1.2). This is a significant improvement over our previous dv-SNARG when in the high-soundness regime (e.g., when we think of $\tau = 80$ and the size of a group element and a hash being 256 bits each). See Figure 1 for an overview of these transformations.

"Fishing in the dark." We believe that our approach can be pushed towards an even smaller SNARG. As previously demonstrated, the packed ElGamal scheme can be used to homomorphically evaluate non-linear functions. However, the class of non-linear functions described in our attack is quite limited. Recall that in the evaluation, the adversary forces a synchronization error of DDL to get non-linear behavior, meaning that $c_i g^{\mathsf{DDL}(c_i)}$ (which is run on the adversary side) is different from $c_0^x g^{\mathsf{DDL}(c_0^x)}$ (which the decryptor computes). In the hash-verified approach, we added a hash to stop such behavior.

We conjecture that it is possible to "integrate" the hash check into the ciphertext, thus making it hard for an adversary to maul the ciphertexts even without the additional cost of the hash output:

- $\mathsf{Compress}(\mathsf{ct} = (c_0, c_1, \dots, c_q))$:
  1. Let $v_i \leftarrow \mathsf{DDL}(c_i)$ for $i \in [q]$.
  2. Let $k_1, \dots, k_q \leftarrow \mathsf{H}(c_1 g^{v_1}, \dots, c_q g^{v_q})$.
  3. Output $(c_0, v_1 + k_1 \mod p, \dots, v_q + k_q \mod p)$.
- $\mathsf{Dec}(\mathsf{sk} = (x_1, \dots, x_q), \mathsf{cct} = (c, e_1, \dots, e_q))$:
  1. Let $v'_i \leftarrow \mathsf{DDL}(c^{x_i})$ for $i \in [q]$.
  2. Let $k_1, \dots, k_n \leftarrow \mathsf{H}(c^{x_1} g^{v'_1}, \dots, c^{x_n} g^{v'_q})$.
  3. Otherwise, output $((v'_1 - e_1 - k_1) \mod p, \dots, (v'_q - e_n - k_q) \mod p)$

While we are currently unable to prove this, intuitively, this scheme should still be secure when combined with a linear PCP over $\mathbb{Z}_p$. To see why, suppose that the decryptor's outputs of $c^{x_i} g^{\mathsf{DDL}(c^{x_i})}$ have high entropy from the perspective of the adversary. In this case, it cannot query $\mathsf{H}(c^{x_1}g^{v'_1}, \ldots, c^{x_n}g^{v'_n})$, and the decryption output will seem truly random. Thus, this attack reduces to a *random* attack function, i.e., a random attack on the underlying PCP (to which all natural linear PCPs are secure).

If, however, the values $c^{x_i}g^{\mathsf{DDL}(c^{x_i})}$ have relatively low entropy from the perspective of the adversary, then each possible input to the hash function H defines an affine function over $\mathbb{Z}_p$. Our linear PCP will then provide soundness against each of the affine functions individually. However, we cannot afford a union bound over all such affine functions. We believe that our scheme is secure since the prover does not have full control of these functions, as they are partially defined using the hash function.

Thus, in this approach, in either case, the prover must "fishing in the dark" for a random function with which to attack the scheme. If we are correct, then the resultant dv-SNARG could have length that approaches 1 group element and 2τ bits. This improvement would be the first step towards proving Conjecture 1.3. We leave further analysis of this encryption scheme when used to compile a SNARG for exciting future work.

3 Preliminaries

For a relation $\mathcal{R} := \{(x, w)\}$, we let $\mathcal{L}(\mathcal{R}) := \{x \mid \exists w,\ (x, w) \in \mathcal{R}\}$. For a vector $\mathbf{a} \in \mathbb{F}^\ell$, we let $\mathbf{a}[i]$ be the i-th entry. For the set $\{n, n+1, \ldots, m\}$ we write $[n, m]$, and abbreviate $[1, m]$ with $[m]$.

We equate prime order fields $\mathbb{F}_p$ with the congruence class $\mathbb{Z}_p$. For $a \in \mathbb{Z}_p$ we define the absolute value $|a|$ to be the minimun distance of an element in the congruence class of a to 0, more specifically $|a| = \min\{|b| \mid b \in \mathbb{Z}, b \mod p = a \mod p\}$. Further we define an operation that lifts $a \in \mathbb{Z}_p$ to the integers $\mathbb{Z}(a) = \min_{|b|}\{b \mid b \in \mathbb{Z}, b \mod p = a \mod p\}$ and similarly an operation that moves a to some $\mathbb{Z}_{p'}$, namely $\mathbb{Z}_{p'}(a) = \{b \mid \mathbb{Z}(a) \mod p' = b \mod p'\}$. Notice, if $p' > p$ then for $a \in \mathbb{Z}_p$ we have $a = \mathbb{Z}_p(\mathbb{Z}_{p'}(a))$.

We use Hoeffding's inequality:

Theorem 3.1. *Let $X_1, \ldots, X_n$ be independent random variables where $X_i \in [-B, B]$ for $B > 0$ and let $X = \sum_i X_i$. Then for every $t > 0$,*

$$\Pr[|X - \mathbb{E}[X]| > t] < 2 \cdot \exp\left(-\frac{t^2}{2 \cdot n \cdot B^2}\right) .$$

3.1 Generic group model

We use Shoup's [Sho97] version of the generic group model.

Definition 3.2. The generic group model models cryptographic group operations via an oracle. For a prime p' the oracle holds a permutation $f : \mathcal{L} \mapsto \mathbb{Z}'_p$ that maps from the label space $\mathcal{L}$ which are just binary representations of the numbers $0, \ldots, p' - 1$, to the group $(\mathbb{Z}'_p, +)$. At the beginning of a security game f is sampled uniformly at random from all permutations and all parties are provided with the label $g \leftarrow f^{-1}(1)$. Throughout security games the parties have oracle access to the oracle $\mathcal{G}$, which take two labels χ_1, χ_2 as input and responds with $f^{-1}(f(\chi_1) + f(\chi_2))$. We denote by $\mathcal{G} \leftarrow \mathsf{GGM}(\lambda)$ the act of sampling a random GGM oracle of size $\geq 2^\lambda$.

Remark 3.3. Our constructions in Construction 4.9 and Corollary 7.2 additionally use a random oracle H, which is a uniformly random function with specified output length χ bits, sampled at the same time as the GGM oracle. For simplicity, in our constructions we always set $\chi = \lambda$ and always count total oracle calls to both oracles.

If an algorithm P has access to an oracle $\mathcal{O}$ we denote this by $P^{\mathcal{O}}$. Further, $y \overset{\mathsf{tr}}{\leftarrow} P^{\mathcal{O}}(x)$ means y is the output of evaluating the algorithm $P^{\mathcal{O}}$ on x and tr is the trace of P's interactions with the oracle $\mathcal{O}$, i.e., it contains all the queries to the oracle and its responses.

3.2 Designated-verifier SNARGs

Definition 3.4. A designated-verifier succinct non-interactive argument (dv-SNARG) in the generic group model, $(\mathsf{Setup}, \mathbf{P}, \mathbf{V})$ for a relation $\mathcal{R} = \{(x, w)\}$ is defined as follows:

- *Syntax.* We describe a dv-SNARG with message length ℓ:
  - The setup algorithm Setup receives an input size parameter 1^n. It outputs a common reference string crs and a verification state st.
  - The (honest) prover algorithm $\mathbf{P}$ receives as input a common reference string crs, instance $x \in \{0,1\}^n$, and witness $w \in \{0,1\}^m$. It outputs a proof $\mathsf{pf} \in \{0,1\}^\ell$.
  - The verifier algorithm $\mathbf{V}$ receives as input a verification state st, and instance $x \in \{0,1\}^n$, and a proof $\mathsf{pf} \in \{0,1\}^\ell$. It outputs a bit $b \in \{0,1\}$.
- *Completeness.* A dv-SNARG has completeness error α if for all $(x, w) \in \mathcal{R}$ and $\lambda \in \mathbb{N}$:

$$\Pr\left[\mathbf{V}^{\mathcal{G}}(\mathsf{st}, x, \mathsf{pf}) = 1 \;\middle|\; \begin{array}{r} \mathcal{G} \leftarrow \mathsf{GGM}(\lambda) \\ (\mathsf{crs}, \mathsf{st}) \leftarrow \mathsf{Setup}^{\mathcal{G}}(1^{|x|}) \\ \mathsf{pf} \leftarrow \mathbf{P}^{\mathcal{G}}(\mathsf{crs}, x, w) \end{array}\right] \geq 1 - \alpha(\lambda, |x|) \ .$$

- *Soundness.* A dv-SNARG has (adaptive) soundness with error δ if for every $\lambda \in \mathbb{N}$, $n \in \mathbb{N}$, and prover $\mathbf{P}'$ that makes at most t queries to the GGM oracle:

$$\Pr\left[\begin{array}{c} x \notin \mathcal{L}(\mathcal{R}) \\ \wedge\ \mathbf{V}^{\mathcal{G}}(\mathsf{st}, x, \mathsf{pf}) = 1 \end{array} \;\middle|\; \begin{array}{r} \mathcal{G} \leftarrow \mathsf{GGM}(\lambda) \\ (\mathsf{crs}, \mathsf{st}) \leftarrow \mathsf{Setup}^{\mathcal{G}}(1^{|x|}) \\ (x, \mathsf{pf}) \leftarrow \mathbf{P}'^{\mathcal{G}}(\mathsf{crs}) \end{array}\right] \leq \delta(\lambda, |x|, t) \ .$$

- *Succinctness.* For every large enough λ, GGM oracle $\mathcal{G}$ in the image of $\mathsf{GGM}(\lambda)$, $(x, w) \in \mathcal{R}$ and $(\mathsf{crs}, \mathsf{st})$ in the image of $\mathsf{Setup}^{\mathcal{G}}(1^{|x|})$, we have $|\mathsf{pf}| = o(w)$ for $\mathsf{pf} = \mathbf{P}^{\mathcal{G}}(\mathsf{crs}, x, w)$.

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

- *(Straight-line) Knowledge soundness.* A dv-SNARG has (adaptive) knowledge soundness (in which case we refer to it as a dv-SNARK) with knowledge κ if there exists an expected polynomial time PPT extractor Ext so that for every $\lambda \in \mathbb{N}$, $n \in \mathbb{N}$, and prover $\mathbf{P}'$ that makes at most t queries to the GGM oracle,

$$\Pr\left[\begin{array}{c} (x, w) \notin \mathcal{R} \\ \wedge\ \mathbf{V}^{\mathcal{G}}(\mathsf{st}, x, \mathsf{pf}) = 1 \end{array} \;\middle|\; \begin{array}{r} \mathcal{G} \leftarrow \mathsf{GGM}(\lambda) \\ (\mathsf{crs}, \mathsf{st}) \leftarrow \mathsf{Setup}^{\mathcal{G}}(1^n) \\ (x, \mathsf{pf}) \overset{\mathsf{tr}}{\leftarrow} \mathbf{P}'^{\mathcal{G}}(\mathsf{crs}) \\ w \leftarrow \mathsf{Ext}(x, \mathsf{pf}, \mathsf{tr}) \end{array}\right] \leq \kappa(\lambda, n, t) \ .$$

3.3 Linear PCPs and strong linear MIPs

A linear PCP is a PCP system where the verifier has query access to (affine) linear functions of the PCP proof.

Definition 3.5. A linear PCP $(\mathbf{P}, (V_{\scriptscriptstyle Q}, V_{\scriptscriptstyle D}))$ for a relation $\mathcal{R} = \{(x, w)\}$ over a finite field $\mathbb{F}$ is defined as follows:

- *Syntax.* We describe a linear PCP with input length n, proof length ℓ, and query complexity q:
  - The verifier query algorithm $V_{\scriptscriptstyle Q}$ receives as input $x \in \mathbb{F}^n$. It outputs a state $\mathsf{st} \in \{0,1\}^*$, and q queries $\mathbf{a}_1, \ldots, \mathbf{a}_q \in \mathbb{F}^\ell$.
  - The (honest) prover algorithm $\mathbf{P}$ receives an input $x \in \mathbb{F}^n$ and witness $w \in \mathbb{F}^h$. It outputs a proof $\pi \in \mathbb{F}^\ell$.

  - The verifier decision algorithm V_D receives as input a state $\mathsf{st} \in \{0,1\}^*$, an input $x \in \mathbb{F}^n$, and query answers $b_1, \ldots, b_q \in \mathbb{F}$. It outputs a bit $b \in \{0,1\}$.

- *Perfect completeness.* A linear PCP has perfect completeness if for all $(x, w) \in \mathcal{R}$:

$$\Pr\left[V_D\left(\mathsf{st}, x, b_1, \ldots, b_q\right) = 1 \;\middle|\; \begin{array}{r} \pi \leftarrow \mathbf{P}(x, w) \\ (\mathsf{st}, \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_Q(x) \\ \forall i \in [q],\ b_i = \langle \pi, \mathbf{a}_i \rangle \end{array} \right] = 1 \ .$$

- *Soundness.* A linear PCP has soundness error (against affine strategies) δ if for every $x \notin \mathcal{L}(\mathcal{R})$, $\pi \in \mathbb{F}^\ell$, and $c_1, \ldots, c_q \in \mathbb{F}$:

$$\Pr\left[V_D\left(\mathsf{st}, x, b_1, \ldots, b_q\right) = 1 \;\middle|\; \begin{array}{c} (\mathsf{st}, \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_Q(x) \\ \forall i \in [q],\ b_i = \langle \pi, \mathbf{a}_i \rangle + c_i \end{array} \right] \leq \delta \ .$$

- *Knowledge.* A linear PCP satisfies knowledge soundness κ if there exists a PPT extractor Ext such that for every x, $\pi \in \mathbb{F}^\ell$, and $c_1, \ldots, c_q \in \mathbb{F}$ if,

$$\Pr\left[V_D\left(\mathsf{st}, x, b_1, \ldots, b_q\right) = 1 \;\middle|\; \begin{array}{c} (\mathsf{st}, \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_Q(x) \\ \forall i \in [q],\ b_i = \langle \pi, \mathbf{a}_i \rangle + c_i \end{array} \right] > \kappa \ ,$$

  then $(x, \mathsf{Ext}(x, \pi, c_1, \ldots, c_q)) \in \mathcal{R}$.

We say that a linear PCP is **smooth** if every query $\mathbf{a}_i$ is 1-wise uniform over $\mathbb{F}^\ell$, and we say that it is **instance-independent** if $V_Q(x)$ is a function only of $|x|$, in which case we specify its input by $1^{|x|}$ (i.e., the verifier query algorithm is $V_Q(1^{|x|})$).

We additionally consider a strong variant of linear MIPs, where the honest prover is restricted to a single linear function, while the malicious adversary can reply to any query with an arbitrary (stateless) function.

Definition 3.6. A strong linear MIP $(\mathbf{P}, (V_Q, V_D))$ for a relation $\mathcal{R} = \{(x, w)\}$ over a finite field $\mathbb{F}$ is defined as follows:

- *Syntax.* We describe a linear PCP with input length n, proof length ℓ, and query complexity q:
  - The verifier query algorithm V_Q receives as input $x \in \mathbb{F}^n$. It outputs a state $\mathsf{st} \in \{0,1\}^*$, and q queries $\mathbf{a}_1, \ldots, \mathbf{a}_q \in \mathbb{F}^\ell$.
  - The (honest) prover algorithm $\mathbf{P}$ receives an input $x \in \mathbb{F}^n$ and witness $w \in \mathbb{F}^h$. It outputs a proof $\pi \in \mathbb{F}^\ell$.
  - The verifier decision algorithm V_D receives as input a state $\mathsf{st} \in \{0,1\}^*$, an input $x \in \mathbb{F}^n$, and query answers $b_1, \ldots, b_q \in \mathbb{F}$. It outputs a bit $b \in \{0,1\}$.

- *Perfect completeness.* A linear PCP has perfect completeness if for all $(x, w) \in \mathcal{R}$:

$$\Pr\left[V_D\left(\mathsf{st}, x, b_1, \ldots, b_q\right) = 1 \;\middle|\; \begin{array}{r} \pi \leftarrow \mathbf{P}(x, w) \\ (\mathsf{st}, \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_Q(x) \\ \forall i \in [q],\ b_i = \langle \pi, \mathbf{a}_i \rangle \end{array} \right] = 1 \ .$$

- *Soundness.* A q-query strong linear MIP has soundness error δ if for every $x \notin \mathcal{L}(\mathcal{R})$, and functions $f_1, \ldots, f_q$:

$$\Pr\left[V_D\left(\mathsf{st}, x, b_1, \ldots, b_q\right) = 1 \;\middle|\; \begin{array}{c} (\mathsf{st}, \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_Q(x) \\ \forall i \in [q],\ b_i = f_i(\mathbf{a}_i) \end{array} \right] \leq \delta \ .$$

- *Knowledge.* A q-query strong linear MIP has knowledge soundness κ if there exists a an expected polynomial-time oracle-aided extractor Ext such that for every x and every set of functions $f_1, \ldots, f_q$ if
$$\Pr\left[V_D(\mathsf{st}, x, b_1, \ldots, b_q) = 1 \;\middle|\; \begin{array}{r} (\mathsf{st}, \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_Q(x) \\ b_i \leftarrow f_i(\mathbf{a}_i) \end{array}\right] > \kappa \ ,$$
then $(x, \mathsf{Ext}^{f_1,\ldots,f_q}(x)) \in \mathcal{R}$.

As with linear PCPs, a linear MIP is is **instance-independent** if $V_Q(x)$ is a function only of $|x|$, in which case we specify its input by $1^{|x|}$ (i.e., the verifier query algorithm is $V_Q(1^{|x|})$).

We also consider bounded variants of PCPs and (strong) MIPs:

Definition 3.7. A q-query LPCP (resp. strong LMIP) $(\mathbf{P}, (V_Q, V_D))$ over a relation $\mathcal{R}$ and finite field $\mathbb{F}_p$ with proof length ℓ is B*-bounded* with error α if for all $(x, w) \in \mathcal{R}$:
$$\Pr\left[b_1, \ldots, b_q \in [-B, B] \;\middle|\; \begin{array}{r} \pi \leftarrow \mathbf{P}(x, w) \\ (\mathsf{st}, \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_Q(x) \\ \forall i \in [q],\ b_i = \langle \mathbb{Z}(\pi), \mathbb{Z}(\mathbf{a}_i) \rangle \in \mathbb{Z} \end{array}\right] \geq 1 - \alpha \ .$$

Note that any LPCP is $(p^2 \cdot \ell)$-bounded with error 0, in which case we either say that it is *trivially bounded.* Whenever we do not give an explicit bound, the LPCP is assumed to be trivially bounded.

Furthermore, observe that (by the union bound) if an LPCP is B-bounded with error α, then its t-wise repetition is B-bounded with error $t \cdot \alpha$.

3.3.1 Linear PCPs used in this paper

In this paper, we us the following linear PCPs known in the literature: For our results, we utilize the existence of the following linear PCPs for arithmetic circuits:

Theorem 3.8 ([BIOW20], Appendix B.1)**.** *Let $C\colon \mathbb{F}_p^n \times \mathbb{F}_p^h \to \mathbb{F}_p$ be an arithmetic circuit of size s over finite field $\mathbb{F}_p$. There exists a 2-query instance-oblivious LPCP over $\mathbb{F}_p$ for $\mathcal{R}_C = \big\{(x, w) \in \mathbb{F}^n \times \mathbb{F}^h \mid C(x, w) = 1\big\}$ with perfect completeness, knowledge soundness error $2/p$ against affine strategies, and proof length $s + s^2$ (field elements).*

If we restrict to Boolean circuits, then there is an LPCP with the same parameters which for any $\lambda \in \mathbb{N}$ with is $O(p^2 s\lambda)$-bounded with error $2^{-\lambda}$.

Theorem 3.9 ([BHI+24], Theorem 1.2)**.** *Let $C\colon \mathbb{F}_p^n \times \mathbb{F}_p^h \to \mathbb{F}_p$ be an arithmetic circuit of size s over finite field $\mathbb{F}_p$ with $p > 2$. There exists a 1-query instance-oblivious LPCP over $\mathbb{F}_p$ for te relation $\mathcal{R}_C = \big\{(x, w) \in \mathbb{F}^n \times \mathbb{F}^h \mid C(x, w) = 1\big\}$ with perfect completeness, knowledge soundness error $O(1/\sqrt{p})$ against affine strategies, and proof length $s \cdot \mathsf{poly}(p)$ (field elements).*

By applying a transformation given in [IKO07, Section 5] from LPCPs to strong LMIPs to Theorem 3.9 we get the following:

Corollary 3.10. *Let $C\colon \mathbb{F}_p^n \times \mathbb{F}_p^h \to \mathbb{F}_p$ be an arithmetic circuit of size s over finite field $\mathbb{F}_p$ with $p > 2$. There exists a $O(1)$-query instance-oblivious strong LMIP over $\mathbb{F}_p$ for $\mathcal{R}_C = \big\{(x, w) \in \mathbb{F}^n \times \mathbb{F}^h \mid C(x, w) = 1\big\}$ with perfect completeness, knowledge soundness error $O(1)$ against affine strategies, and proof length $O(s \cdot \mathsf{poly}(p))$ (field elements).*

In Section 6.1 we give an alternate transformation from LPCP to strong LMIP which has better concrete parameters, which we apply to the LPCP described in Theorem 3.8.

Following the discussion on open problems in Section 1.2, we add a conjecture that the LPCP of Theorem 3.9 is bounded. We stress that this conjecture is included here purely to formally define what we mean in the discussion, and is not used in this paper.

Conjecture 3.11. The LPCP from Theorem 3.9 is $O(\lambda p^2 \sqrt{s})$-bounded with error $2^{-\lambda}$.

3.4 Linearity testing

In our construction of strong linear MIPs (Section 6.1) we utilize a variant of the [BLR93] linearity test for linear-consistent functions.

Definition 3.12. A triple of functions $f_1, f_2, f_3 \colon \mathbb{F}^\ell \to \mathbb{F}$ is *linear-consistent* if there exists a linear function $g \colon \mathbb{F}^\ell \to \mathbb{F}$ and $c_1, c_2, c_3 \in \mathbb{F}$ so that $c_1 + c_2 = c_3$ and for every $i \in [3]$ and $z \in \mathbb{F}^\ell$, $f_i(z) = g(z) + c_i$.

Theorem 3.13 ([AHRS01], Theorem 2)**.** *Let $f_1, f_2, f_3 \colon \mathbb{F}^\ell \to \mathbb{F}$. If*

$$\delta := \Pr_{z_1, z_2 \leftarrow \mathbb{F}^\ell} [f_1(z_1) + f_2(z_2) \neq f_3(z_1 + z_2)] < \frac{2}{9} \ ,$$

then there exist a triple of linear-consistent functions $g_1, g_2, g_3 \colon \mathbb{F}^\ell \to \mathbb{F}$ so that for every $i \in [3]$, $\Delta(f_i, g_i) \leq \delta$.

3.5 Distributed discrete log

In our packed ElGamal encryption schemes (Section 4) use the distributed discrete log algorithm:

Lemma 3.14 ([BGI16, BGI17])**.** *Let $\delta > 0$, $B \in \mathbb{N}$, p' be a prime with $B, T < p'$, and $\delta = 4B/T$. There exists an algorithm DDL that does T GGM queries such that for all GGM labels $h \in \mathcal{L}$,*

$$\Pr\left[\forall x \in [-B, B], \begin{array}{r} \mathsf{DDL}^{\mathcal{G}}_{B,\delta}(h) \\ - \mathsf{DDL}^{\mathcal{G}}_{B,\delta}(h \cdot g^x) = x \end{array} \middle| \ \mathcal{G} \leftarrow \mathsf{GGM}(p') \right] \geq 1 - \delta \ .$$

Further, $\forall x \notin [-T, T]$ we have

$$h \cdot g^{\mathsf{DDL}^{\mathcal{G}}_{B,\delta}(h)} \neq h \cdot g^{x + \mathsf{DDL}^{\mathcal{G}}_{B,\delta}(h \cdot g^x)}.$$

Remark 3.15. Previous work on distributed discrete logarithms are in the plain model. They require a large group and a function that maps elements of this group to random bit-strings. The generic group model provides both of these properties.

4 Compressible encryptions schemes

Our constructions of designated-verifier SNARGs will utilize linearly homomorphic encryption schemes that have compressible ciphertexts. In this section, we define compressible encryption schemes. In subsequent sections we give constructions of such schemes: in Section 4.1 we describe the packed ElGamal encryption scheme, and in Section 4.2 we show a variant on this scheme that additionally uses a hash function.

Definition 4.1 (Compressible linearly homomorphic encryption)**.** A compressible bounded linearly homomorphic encryption scheme in the generic group model with bounded message space and compressed ciphertext size σ_{cct} is a tuple of algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval}, \mathsf{Compress})$ that must satisfy the following properties:

- *Syntax.* We describe an encryption scheme with homomorphism modulus p', $n \in \mathbb{N}$ slots, plaintext moduli $p_1, \ldots, p_n$, decryption bound B, ciphertext size σ_{ct}, compressed ciphertext size σ_{cct}, encryption, decryption, and eval running times $t_{\mathsf{enc}}, t_{\mathsf{dec}}, t_{\mathsf{eval}}$. All algorithms have access to a GGM oracle of size λ.
  - KeyGen: Outputs a public key pk and a secret key sk.
  - $\mathsf{Enc}(\mathsf{pk}, m)$: On input a public key pk and a message $m \in \mathbb{Z}^n_{p'}$, outputs a ciphertext ct. This is done in time $t_{\mathsf{enc}}(\lambda)$.
  - $\mathsf{Dec}(\mathsf{sk}, \mathsf{cct})$: On input a secret key sk and a compressed ciphertext cct, output a message $m \in \mathbb{Z}_{p_1} \times \cdots \times \mathbb{Z}_{p_n} \cup \{\perp\}$. This is done in time $t_{\mathsf{dec}}(\lambda)$.

  - $\mathsf{Eval}(\mathsf{pk}, \mathsf{ct}_1, \dots, \mathsf{ct}_\ell, \pi)$: On input a public key pk, ciphertexts $\mathsf{ct}_1, \dots, \mathsf{ct}_\ell$ and a linear function $\pi \in \mathbb{Z}_{p'}^n$, outputs a new ciphertext ct'. This is done in time $t_{\mathsf{eval}}(\lambda, \ell)$.
  - $\mathsf{Compress}(\mathsf{pk}, \mathsf{ct})$: On input a public key pk and a ciphertext ct, outputs a compressed ciphertext cct with $|\mathsf{cct}| = \sigma_{\mathsf{cct}}$.

- *Correctness.* The encryption has correctness error $\varepsilon_{\mathsf{cor}}$ if for every $\lambda, \in \mathbb{N}$, $\mathbf{m}_1, ..., \mathbf{m}_t \in \mathbb{Z}_{p'}^n$, and $\pi \in \mathbb{Z}_{p'}^\ell$

$$\Pr\left[\mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct}) = \left(\mathbb{Z}_{p_i}\left(\langle \pi, \begin{pmatrix} \mathbf{m}_1[i] \\ \vdots \\ \mathbf{m}_t[i] \end{pmatrix} \rangle \right) \right)_{i \in [n]} \;\middle|\; \begin{array}{r} \mathcal{G} \leftarrow \mathsf{GGM}(\lambda) \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}} \\ \forall i \in [t],\ \mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, \mathbf{m}_i) \\ \mathsf{ct}' \leftarrow \mathsf{Eval}^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \dots, \mathsf{ct}_\ell, \pi) \\ \mathsf{cct} \leftarrow \mathsf{Compress}^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}') \\ \forall i \in [n], \langle \pi, \begin{pmatrix} \mathbf{m}_1[i] \\ \vdots \\ \mathbf{m}_t[i] \end{pmatrix} \rangle \in [-B, B] \end{array} \right] \geq 1 - \varepsilon_{\mathsf{cor}}(\lambda) \ .$$

  We say that it is correct if $\varepsilon_{\mathsf{cor}}(\lambda) = \mathsf{negl}(\lambda)$.

- *Semantic security.* The encryption scheme has semantic security advantage $\varepsilon_{\mathsf{sem}}$ for t-query adversaries if for any $\lambda \in \mathbb{N}$ and adversary $\mathcal{A}$ that makes at most t queries to the GGM oracle:

$$\Pr\left[\begin{array}{c} \forall i,\ m_{0,i}, m_{1,i} \in \mathbb{Z}_q^n \\ \wedge\ b = b' \end{array} \;\middle|\; \begin{array}{r} \mathcal{G} \leftarrow \mathsf{GGM}(\lambda) \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}} \\ ((m_{0,1}, \dots, m_{0,\ell}), (m_{1,1}, \dots, m_{1,\ell}), \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}^{\mathcal{G}}(\mathsf{pk}) \\ b \leftarrow \{0, 1\} \\ \mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, m_{b,i}) \\ b' \leftarrow \mathcal{A}^{\mathcal{G}}(\mathsf{ct}_1, \dots, \mathsf{ct}_\ell, \mathsf{st}_{\mathcal{A}}) \end{array} \right] \leq \frac{1}{2} + \varepsilon_{\mathsf{sem}}(\lambda, t, \ell) \ .$$

  We say that it is semantically secure if for $t, \ell = \mathsf{poly}(\lambda)$ we have $\varepsilon_{\mathsf{sem}}(\lambda, t, \ell) = \mathsf{negl}(\lambda)$.

Remark 4.2. Construction 4.9 has an additional random oracle with output length λ. It is straightforward to add sampling of the random oracle into the notation above.

4.1 Packed ElGamal

We describe the packed ElGamal compressible encryption scheme:

Theorem 4.3. *The packed ElGamal encryption scheme described in Construction 4.4 is compressible bounded linearly homomorphic with the following properties:*

- *Homomorphism modulus: p', the size of the generic group,*
- *Number of Slots: n,*
- *Correctness error: 0,*
- *Semantic security advantage: $\varepsilon_{\mathsf{sem}}(\lambda, t, \ell) = 4t^2/2^\lambda$.*
- *Plaintext moduli: $p = p_1 = ... = p_n$,*
- *Ciphertext size: $n + 1$ $\mathbb{G}$-elements,*
- *Compressed ciphertext size: 1 $\mathbb{G}$-element and $\lceil n \log p \rceil$ bits,*
- *Encryption time: $(4n + 2) \log \lambda$ group operations,*
- *Evaluation time for ℓ linear combination: $\ell(2 \log \lambda + 1)(n + 1)$ group operations,*
- *Decryption time: $n(8Bn + \log \lambda)$ group operations,*
- *Expected compression time: $32Bn^2 + 2(n + 1) \log \lambda$ group operations.*

Construction 4.4 (Packed ElGamal)**.** We specify the encryption scheme, parameterized by a message-space $p, B, n \in \mathbb{N}$ and number of supported additions ℓ. Let $\delta = 1/2n$ and number of GGM queries per DDL $T = 8Bn$.

$\mathsf{KeyGen}^{\mathcal{G}}$:

1. Let p' be the order of the generic group $\mathcal{G}$ and g its generator.
2. For every $i \in [n]$, sample $x_i \leftarrow_{\$} \mathbb{Z}_{p'}$.
3. Output $\mathsf{pk} = (g^{x_1}, \dots, g^{x_n})$ and $\mathsf{sk} = (x_1, \dots, x_n)$.

$\mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, m)$:

1. Parse $\mathsf{pk} = (h_1, \dots, h_n)$ and $m \in \mathbb{Z}_{p'}$.
2. Sample $r \leftarrow_{\$} \mathbb{Z}_{p'}$.
3. Output $\mathsf{ct} = (g^r, h_1^r \cdot g^{m_1}, \dots, h_n^r \cdot g^{m_n})$.

$\mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct})$:

1. Parse $\mathsf{sk} = (x_1, \dots, x_n)$ and $\mathsf{cct} = (c, e_1, \dots, e_n)$.
2. For every $i \in [n]$, let $m_i = (\mathsf{DDL}_{B,\delta}(c^{x_i}) - e_i) \mod p$.
3. Output $m = (m_1, \dots, m_n)$.

$\mathsf{Eval}^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \dots, \mathsf{ct}_t, \pi)$:

1. Parse $\mathsf{pk} = (h_1, \dots, h_n)$ and $\mathsf{ct}_i = (a_i, \mathbf{b}_i)$.
2. Let $a' = a_1^{\pi_1} \cdot \dots \cdot a_\ell^{\pi_\ell}$ and $b'_j = \mathbf{b}_1[j]^{\pi_1} \cdot \dots \cdot \mathbf{b}_\ell[j]^{\pi_\ell}$ for $j \in [n]$.
3. Output $\mathsf{ct}' = (a', b'_1, \dots, b'_n)$.

$\mathsf{Compress}^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct})$:

1. Parse $\mathsf{pk} = (h_1, \dots, h_n)$ and $\mathsf{ct} = (a, b_1, \dots, b_n)$.
2. Do in a loop:
   - Sample $r \leftarrow_{\$} \mathbb{Z}_{p'}$ uniformly at random.
   - Compute $a \leftarrow a \cdot g^r$, and $b_i \leftarrow b_i \cdot h_i^r$ for every $i \in [n]$.
3. Until for every $i \in [n]$ and $j \in [-B, B]$ it holds that

$$\mathsf{DDL}^{\mathcal{G}}_{B,\delta}(b_i \cdot g^j) + j = \mathsf{DDL}^{\mathcal{G}}_{B,\delta}(b_i).$$

4. Output $\mathsf{cct} = (a, \mathsf{DDL}^{\mathcal{G}}_{B,\delta}(b_1) \mod p, \dots, \mathsf{DDL}^{\mathcal{G}}_{B,\delta}(b_n) \mod p)$.

Lemma 4.5. *Construction 4.4 satisfies correctness and the running times are as described in Theorem 4.3.*

Proof. Fix parameters $\lambda, p, B, n, t \in \mathbb{N}$, messages $m_1, \dots, m_t \in \mathbb{Z}_p^n$, and $\pi \in \mathbb{Z}_p^t$. We follow the correctness experiment, keeping track of what each value is:

- *Setup.* Fix any $(\mathsf{pk}, \mathsf{sk})$ sampled by KeyGen. Then letting $\mathsf{sk} = (x_1, \dots, x_n)$, we have $\mathsf{pk} = (g^{x_1}, \dots, g^{x_n})$.
- *Encryption.* For every $i \in [\ell]$, and any randomness r_i sampled during the encryption of ct_i, and $\mathbf{m}_i$, it holds that

$$\begin{aligned}\mathsf{ct}_i &= (g^{r_i}, h_1^{r_i} \cdot g^{\mathbf{m}_i[1]}, \dots, h_n^{r_i} \cdot g^{\mathbf{m}_i[n]}) \\ &= (g^{r_i}, g^{x_1 \cdot r_i + \mathbf{m}_i[1]}, \dots, g^{x_n \cdot r_i + \mathbf{m}_i[n]}) = (a_i, \mathbf{b}_i) \ .\end{aligned}$$

- *Linear evaluation.* Following the evaluation step, we have $\mathsf{ct}' = (a', b'_1, \dots, b'_n)$ where

$$a' = \prod_{i \in [\ell]} a_i^{\pi_i} = g^{\sum_{i \in [\ell]} \pi_i \cdot r_i} \quad ,$$

  and for every $k \in [n]$,

$$b'_k = \prod_{i \in [\ell]} \mathbf{b}_i[k]^{\pi_i} = g^{\sum_{i \in [\ell]} \pi_i \cdot r_i \cdot x_k + \pi_i \cdot \mathbf{m}_i[k]} \quad .$$

- *Compression.* The compressed ciphertext is $\mathsf{cct} = (c, v_1, \ldots, v_n)$, where for some $r \in \mathbb{Z}'_p$ we have

$$c = a' \cdot g^r = g^{r + \sum_{i \in [\ell]} \pi_i \cdot r_i} \ ,$$

and for every $k \in [n]$,

$$v_k = \mathsf{DDL}(b'_k \cdot h_k^r) = \mathsf{DDL}\left(g^{r \cdot x_k + \sum_{i \in \ell} \pi_i \cdot r_i \cdot x_k + \pi_i \cdot \mathbf{m}_i[k]}\right)$$

and we know that for all $j \in [-B, B]$ we have

$$\begin{aligned} &\mathsf{DDL}\left(g^{r \cdot x_k + \sum_{i \in \ell} \pi_i \cdot r_i \cdot x_k + \pi_i \cdot \mathbf{m}_i[k]}\right) + j \\ =&\mathsf{DDL}\left(g^{r \cdot x_k + \sum_{i \in [\ell]} \pi_i \cdot r_i \cdot x_k + \pi_i \cdot \mathbf{m}_i[k] - j}\right) \\ =&\mathsf{DDL}\left(g^{r \cdot x_k + \sum_{i \in [\ell]} \pi_i \cdot r_i \cdot x_k + \sum_{i \in [\ell]} \pi_i \cdot \mathbf{m}_i[k] - j}\right) \end{aligned}$$

- *Decryption.* For every $k \in [n]$ the decryptor computes

$$\begin{aligned} &(\mathsf{DDL}(c^{x_k}) - v_k) \mod p \\ =&\Big(\mathsf{DDL}\left(g^{(r + \sum_{i \in [\ell]} \pi_i \cdot r_i) \cdot x_k}\right) \\ &- \mathsf{DDL}\left(g^{r \cdot x_k + \sum_{i \in \ell} \pi_i \cdot r_i \cdot x_k + \sum_{i \in [\ell]} \pi_i \cdot \mathbf{m}_i[k]}\right)\Big) \mod p \end{aligned}$$

Because $\sum_{i \in [\ell]} \pi_i \cdot \mathbf{m}_i[k] \in [-B, B]$ we get

$$\begin{aligned} &\Big(\mathsf{DDL}\left(g^{(r + \sum_{i \in [\ell]} \pi_i \cdot r_i) \cdot x_k}\right) \\ &- \mathsf{DDL}\left(g^{r \cdot x_k + \sum_{i \in \ell} \pi_i \cdot r_i \cdot x_k + \sum_{i \in [\ell]} \pi_i \cdot \mathbf{m}_i[k]}\right)\Big) \mod p \\ =&\Big(\mathsf{DDL}\left(g^{r \cdot x_k + \sum_{i \in \ell} \pi_i \cdot r_i \cdot x_k + \sum_{i \in [\ell]} \pi_i \cdot \mathbf{m}_i[k]}\right) + \sum_{i \in [\ell]} \pi_i \cdot \mathbf{m}_i[k] \\ &- \mathsf{DDL}\left(g^{r \cdot x_k + \sum_{i \in \ell} \pi_i \cdot r_i \cdot x_k + \sum_{i \in [\ell]} \pi_i \cdot \mathbf{m}_i[k]}\right)\Big) \mod p \\ =&\left(\sum_{i \in [\ell]} \pi_i \cdot \mathbf{m}_i[k]\right) \mod p \ , \end{aligned}$$

which is the correct value.

What is left is to analyze the running times of the algorithms. For everything except $\mathsf{Compress}$ this is trivial. By Lemma 3.14 we get that for every $i \in [n]$

$$\Pr\left[\forall j \in [-B, B] : \mathsf{DDL}_{B,\delta}(b_i \cdot g^{-j}) = \mathsf{DDL}_{B,\delta}(b_i) - j\right] \geq 1 - \delta$$

Therefore, by union bound it follows that

$$\Pr\left[\forall j \in [-B, B], i \in [n] : \mathsf{DDL}_{B,\delta}(b_i \cdot g^{-j}) = \mathsf{DDL}_{B,\delta}(b_i) - j\right] \geq 1 - n\delta = 1/2$$

Therefore, $\mathsf{Compress}$ runs the loop a constant number of times in expectation.

□

Remark 4.6. As described above the $\mathsf{Compress}$ runs in polynomial time with overwhelming probability. To turn it into strict poly time one can limit the number of times the loop is run. Further, to drastically speed up compression one can leave out the loop entirely. Both of these changes result in imperfect correctness.

Lemma 4.7. *The encryption scheme described in Construction 4.4 satisfies statistical semantic security for multiple ciphertexts in the generic group model against adversaries with t queries with a statistical distance of $4t^2/p' \leq 4t^2/2^\lambda$.*

Proof. Follows from arguments almost identical to Claim 5.4. □

4.2 Packed ElGamal with hash check

We describe the packed ElGamal compressible encryption scheme extended with a hash:

Theorem 4.8. *The packed ElGamal with hash check encryption scheme described in Construction 4.9 is compressible bounded linearly homomorphic with the following properties:*

- *Homomorphism modulus: p', the size of the generic group,*
- *Number of Slots: n,*
- *Correctness error: 0,*
- *Semantic security advantage: $\varepsilon_{\mathsf{sem}}(\lambda, t, \chi, \ell) = 4t^2/2^\lambda$ for big enough t.*
- *Plaintext moduli: $p_1, ..., p_n$,*
- *Ciphertext size: $n+1$ $\mathbb{G}$-elements,*
- *Compressed ciphertext size: 1 $\mathbb{G}$-element, 1 $\mathbb{H}$ hash and $\lceil \sum_{i\in[n]} \log p_i \rceil$ bits,*
- *Encryption time: $(4n+2)\log\lambda$ group operations,*
- *Evaluation time for ℓ linear combination: $\ell(2\log\lambda+1)(n+1)$ group operations,*
- *Decryption time: $n(8Bn+\log\lambda)$ group operations and 1 $\mathbb{H}$ hash operation,*
- *Expected compression time: $32Bn^2 + 2(n+1)\log\lambda$ group operations and 1 $\mathbb{H}$ hash operation.*

Construction 4.9 (Packed ElGamal with hash check)**.** We specify the encryption scheme, parameterized by $n, p_1, \ldots, p_n, B \in \mathbb{N}$ and number of supported additions ℓ. Let $\delta = 1/2n$, number of GGM queries per DDL $T = 8Bn$, and H be a random oracle with output size χ.

$\mathsf{KeyGen}^{\mathcal{G}}$:

```
    1. Let $p'$ be the order of the generic group $\mathcal{G}$ and $g$ its generator.
    2. For every $i \in [n]$, sample $x_i \leftarrow_\$ \mathbb{Z}_{p'}$.
    3. Output $\mathsf{pk} = (g^{x_1}, \ldots, g^{x_n})$ and $\mathsf{sk} = (x_1, \ldots, x_n)$.
```

$\mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, m)$:

```
    1. Parse $\mathsf{pk} = (h_1, \ldots, h_n)$ and $\mathbf{m} \in \mathbb{Z}_{p'}^n$.
    2. Sample $r \leftarrow_\$ \mathbb{Z}_{p'}$.
    3. Output $\mathsf{ct} = (g^r, h_1^r \cdot g^{\mathbf{m}[1]}, \ldots, h_n^r \cdot g^{\mathbf{m}[n]})$.
```

$\mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct})$:

```
    1. Parse $\mathsf{sk} = (x_1, \ldots, x_n)$ and $\mathsf{cct} = (c, e_1, \ldots, e_n, k)$.
    2. For every $i \in [n]$, let $m_i = (\mathsf{DDL}^{\mathcal{G}}_{B,\delta}(c^{x_i}) - e_i) \mod p_i$.
    3. Output $m = (m_1, \ldots, m_n)$.
```

$\mathsf{Eval}^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell, \pi)$:

```
    1. Parse $\mathsf{pk} = (h_1, \ldots, h_n)$ and $\mathsf{ct}_i = (a_i, \mathbf{b}_i)$.
    2. Let $a' = a_1^{\pi_1} \cdot \ldots \cdot a_\ell^{\pi_\ell}$ and $b'_j = \mathbf{b}_1^{\pi_1}[j] \cdot \ldots \cdot \mathbf{b}_\ell^{\pi_\ell}[j]$ for $j \in [n]$.
    3. Output $\mathsf{ct}' = (a', b'_1, \ldots, b'_n)$.
```

$\mathsf{Compress}^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct})$:

```
    1. Parse $\mathsf{pk} = (h_1, \ldots, h_n)$ and $\mathsf{ct} = (a, b_1, \ldots, b_n)$.
    2. Do in a loop:
        • Sample $r \leftarrow_\$ \mathbb{Z}_{p'}$ uniformly at random.
        • Compute $a \leftarrow a \cdot g^r$, and $b_i \leftarrow b_i \cdot h_i^r$ for every $i \in [n]$.
    3. Until for every $i \in [n]$ and $j \in [-B, B]$ it holds that
           $$\mathsf{DDL}^{\mathcal{G}}_{B,\delta}(b_i \cdot g^j) + j = \mathsf{DDL}^{\mathcal{G}}_{B,\delta}(b_i).$$
    4. For $i \in [n]$ let $e_i \leftarrow \mathsf{DDL}^{\mathcal{G}}_{B,\delta}(b_i)$
```

5. Let $k \leftarrow \mathsf{H}(a, e_1, \ldots, e_n)$.
6. Output $\mathsf{cct} = (a, e_1 \mod p_1, \ldots, e_n \mod p_n, k)$.

Lemma 4.10. *The encryption scheme described in Construction 4.9 is correct and statistically semantically secure for multiple ciphertexts in the generic group model with a loss of* $4t^2/p' \leq 4t^2/2^\lambda$*. Moreover, the running times are as described in Theorem 4.8.*

Proof. This follows from the exact same arguments as in Lemmas 4.5 and 4.7. ☐

5 Targeted malleability

In this section we define malleability notions, and prove that our encryption schemes satisfy these notions in the generic group model. In Section 5.1, we define the two notions that we consider: isolated homomorphism, and bound-limited homomorphism. Then, in Section 5.2, we show that the packed ElGamal encryption scheme satisfies isolated homomorphism, and in Section 5.3 we show that packed ElGamal with hash satisfies bound-limited homomorphism.

5.1 Malleability notions

We define two malleability notions for compressible encryption schemes. The first says that no adversary can mix information between different slots of the messages:

Definition 5.1 (Isolated homomorphism)**.** An n-slot compressible encryption scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval}, \mathsf{Compress})$ is *isolated linearly homomorphic* with distinguishing error $\varepsilon_{\mathsf{ih}} = \varepsilon(\lambda, t, m)$ in the generic group model if there exists a poly time simulator $\mathcal{S}$ such that for every plaintext generator $\mathcal{T}$ and oracle machine $\mathcal{A}$ that makes t queries to the GGM oracle, the following distributions have statistical distance at most $\varepsilon_{\mathsf{ih}}$:

- **Real world:**
  1. Sample $\mathcal{G} \leftarrow \mathsf{GGM}(\lambda)$.
  2. Let $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}}$.
  3. $(\mathbf{a}_1, \ldots, \mathbf{a}_m, \mathsf{st}_{\mathcal{T}}) \leftarrow \mathcal{T}(\mathsf{pk})$.
  4. $\mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, \mathbf{a}_i)$ for all $i \in [m]$.
  5. $(\mathsf{cct}, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_m)$.
  6. If $\mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct}) = \perp$ output $\perp$
  7. $(a'_1, \ldots, a'_n) \leftarrow \mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct})$
  8. Output $(\mathsf{st}_{\mathcal{T}}, \mathsf{st}_{\mathcal{A}}, a'_1, \ldots, a'_n)$.
- **Ideal world:**
  1. Sample $\mathcal{G} \leftarrow \mathsf{GGM}(\lambda)$.
  2. Let $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}}$.
  3. $(\mathbf{a}_1, \ldots, \mathbf{a}_m, \mathsf{st}_{\mathcal{T}}) \leftarrow \mathcal{T}(\mathsf{pk})$.
  4. $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbf{a}_i)$ for all $i \in [m]$.
  5. $(\mathsf{cct}, \mathsf{st}_{\mathcal{A}}) \overset{\mathsf{tr}}{\leftarrow} \mathcal{A}^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_m)$, where tr is the trace of queries $\mathcal{A}$ made to the generic group oracle and the oracle's responses.
  6. $(f_1, \ldots, f_n) \leftarrow \mathcal{S}(\mathsf{tr}, \mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_m, \mathsf{cct})$.
  7. If $\mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct}) = \perp$ output $\perp$.
  8. $a'_i = f_i(\mathbf{a}_1[i], \ldots, \mathbf{a}_m[i])$ for all $i \in [n]$.
  9. Output $(\mathsf{st}_{\mathcal{T}}, \mathsf{st}_{\mathcal{A}}, a'_1, \ldots, a'_n)$.

We say that the encryption scheme is isolated homomorphic if $t = \mathsf{poly}(\lambda)$ we have $\varepsilon_{\mathsf{ih}}(\lambda, m, t) = \mathsf{negl}(\lambda)$.

The second notion is a notion of bound-limited linear-only encryption, which says that any adversary can only apply linear functions to an encrypted message. These linear functions may be over a large field $\mathbb{F}_{p'}$, but their results need to be within a bounded range. If they are within that range the decryptor learns the values but modded by some smaller modulus p. This will later match our definition of modded linear PCPs (see Section 6.2).

Definition 5.2 (Bound-limited homomorphism)**.** A compressible encryption scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval}, \mathsf{Compress})$ is *B'-bounded limited homomorphic* with n slots and moduli p', $(p_i)_{i\in[n]}$ with distinguishing error $\varepsilon_{\mathsf{bnd}} = \varepsilon_{\mathsf{bnd}}(\lambda, t, m)$ in the generic group model if there exists a poly time simulator $\mathcal{S}$ such that that for every plaintext generator $\mathcal{T}$ and oracle machine $\mathcal{A}$ makes t to the GGM oracle the following distributions have statistical distance at most $\varepsilon_{\mathsf{bnd}}$:

- **Real world:**
  1. Sample $\mathcal{G} \leftarrow \mathsf{GGM}(\lambda)$.
  2. Let $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}}$.
  3. $(\mathbf{a}_1, \dots, \mathbf{a}_m, \mathsf{st}_{\mathcal{T}}) \leftarrow \mathcal{T}(\mathsf{pk})$.
  4. $\mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, \mathbf{a}_i)$ for all $i \in [m]$.
  5. $(\mathsf{cct}, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \dots, \mathsf{ct}_m)$.
  6. If $\mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct}) = \perp$ output $\perp$
  7. $(a'_1, \dots, a'_n) \leftarrow \mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct})$
  8. Output $(\mathsf{st}_{\mathcal{T}}, \mathsf{st}_{\mathcal{A}}, a'_1, \dots, a'_n)$.
- **Ideal world:**
  1. Sample $\mathcal{G} \leftarrow \mathsf{GGM}(\lambda)$.
  2. Let $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}$.
  3. $(\mathbf{a}_1, \dots, \mathbf{a}_m, \mathsf{st}_{\mathcal{T}}) \leftarrow \mathcal{T}(\mathsf{pk})$.
  4. $\mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, \mathbf{a}_i)$ for all $i \in [m]$.
  5. $(\mathsf{cct}, \mathsf{st}_{\mathcal{A}}) \overset{\mathsf{tr}}{\leftarrow} \mathcal{A}^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \dots, \mathsf{ct}_m)$, where tr is the trace of queries $\mathcal{A}$ made to the generic group oracle and the oracle's responses.
  6. $(\Pi \in \mathbb{Z}_{p'}^m, b_1 \in \mathbb{Z}_{p_1}, \dots, b_1 \in \mathbb{Z}_{p_n}) \leftarrow \mathcal{S}(\mathsf{tr}, \mathsf{pk}, \mathsf{ct}_1, \dots, \mathsf{ct}_m, \mathsf{cct})$.
  7. If $\mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct}) = \perp$ output $\perp$.
  8. $a'_i = \mathbb{Z}_{p_i}\left(\sum_{j\in[m]} \Pi_j \cdot \mathbf{a}_j[i]\right) + b_i$ for all $i \in [n]$.
  9. If there exists an i such that $a'_i \notin [-B', B']$ output $\perp$.
  10. Output $(\mathsf{st}_{\mathcal{T}}, \mathsf{st}_{\mathcal{A}}, a'_1, \dots, a'_n)$.

As with the definition of compressible encryption schemes, here we also allow a random oracle H which is sampled together with $\mathcal{G}$ in Item 1. We say that the encryption scheme is bound-limited homomorphic if $t = \mathsf{poly}(\lambda)$ we have $\varepsilon_{\mathsf{ih}}(\lambda, m, t) = \mathsf{negl}(\lambda)$.

5.2 Isolated homomorphism of packed ElGamal

In this section, we prove that the packed ElGamal encryption scheme is isolated homomorphic.

Lemma 5.3. *Packed ElGamal is isolated homomorphic in the GGM with a distinguishing error $\varepsilon_{\mathsf{ih}}(\lambda, t, m) = 4t^2/p' \le 4t^2/2^\lambda$ for $t > 4t_{\mathsf{Dec}} + 1$, where t_{Dec} is the number of queries the decryption algorithm does.*

Proof. Let p' be the size of the GGM group. We design a simulator for the isolated homomorphism experiment:

1. In the beginning of the security game, the simulator receives the trace tr, public key $\mathsf{pk} := (h_j)_{j\in[n]}$, and $\mathsf{ct}_i := (\mathsf{ct}_{i,0}, (\mathsf{ct}_{i,j})_{j\in[n]})$ for $i \in [m]$. The simulator initializes an empty table T and for $i \in [m]$ and $j \in [n]$ adds the expressions

$$g \mapsto 1, \qquad h_j \mapsto \hat{x}_j, \qquad \mathsf{ct}_{i,0} \mapsto \hat{r}_i, \qquad \text{and} \qquad \mathsf{ct}_{i,j} \mapsto \hat{r}_i \hat{x}_j + \hat{\mathbf{a}}_i[j]$$

to the table where $\hat{x}_j$, $\hat{r}_i$, and $\hat{\mathbf{a}}_i[j]$ are formal variables representing secret key, randomness and messages respectively.

2. The simulator goes through the trace tr first to last entry and does the following for each entry:

   Each entry has two input labels handles ξ_1 and ξ_2, the simulator checks whether there are mappings T from ξ_1 and ξ_2 to polynomials over formal variables Φ_1 and Φ_2 in the table, respectively. If ξ_i does not map to a polynomial in the table T and the existing formal variables are $(\hat{u}_j)_{j\in[\ell]}$ then the simulator generates a new formal variable $\hat{u}_{\ell+1}$ and adds the mapping $\xi_i \mapsto \hat{u}_{\ell+1}$ into the table T. Now, ξ_1 and ξ_2 both have mappings in T to polynomials over formal variables Φ_1, Φ_2.

   The simulator computes the polynomial $\Phi_3 := \Phi_1 + \Phi_2$. The simulator looks at the output label of the trace entry ξ_3. If the table T contains an entry $\xi_3 \mapsto \Phi_3'$ for some polynomial Φ_3' then the simulator outputs $\perp$ if $\Phi_3' \not\equiv \Phi_3$.

3. The simulator also has the adversary's output $\mathsf{cct} := (\mathsf{cct}_0, (e_j')_{j\in[n]})$. The simulator checks whether T contains mappings $\mathsf{cct}_0 \mapsto \Phi_0'$, where Φ_0' is a polynomial over formal variables equivalent to $\sum_{i\in[m]} \alpha_i \hat{r}_i + \beta$ for $\alpha_1, \ldots, \alpha_m, \beta \in \mathbb{Z}_{p'}$. If this is not the case, then for each $j \in [n]$ the simulator samples a uniform label v_j not used in T yet, and outputs $f_1, \ldots, f_n$ functions that entirely ignore the input:

   - $f_j(\mathbf{a}_1[j], \ldots, \mathbf{a}_m[j])$: Output $(\mathsf{DDL}_g(v_j) - e_j) \mod q$.

4. If the simulator has reached this point it outputs the functions $f_1, \ldots, f_n$ defined below. The functions have the labels for $(\mathsf{ct}_j^{(i)})_{i\in[m]}$ hardcoded, which correspond to the polynomials $\hat{r}_i\hat{x}_j + \hat{\mathbf{a}}_i[j]$

   - $f_j(\mathbf{a}_1[j], \ldots, \mathbf{a}_m[j])$:
     (a) Let $u_j \leftarrow \sum_i \alpha_i \mathbf{a}_i[j]$.
     (b) Compute a label v_j for the polynomial over formal variables $\sum_i \alpha_i(\hat{r}_i\hat{x}_j + \hat{\mathbf{a}}_i[j]) + \beta\hat{x}_j - u_j$.
     (c) Output $(\mathsf{DDL}_g(v_j) - e_j) \mod q$.

We show the extracted function outputs the correct distribution if the output ciphertext decrypts successfully with overwhelming probability. We prove this via hybrid argument.

Hyb_0**:** It is the same as the real distribution in Definition 5.1. In more detail:

1. In the beginning of the security game, for each $j \in [n]$ the experiment samples $x_j \leftarrow_\$ \mathbb{Z}_{p'}$. The experiment initializes an empty table T and adds the mapping $g \mapsto 1$. For $j \in [n]$ the experiment checks whether there already exists a mapping from to x_j. If not the experiment samples a new distinct label h_j and adds $h_j \mapsto x_j$ to the table T. It sets the public key $\mathsf{pk} := (h_j)_{j\in[n]}$ and the secret key $\mathsf{sk} := (x_j)_{j\in[n]}$.
2. Then the experiment samples the plaintexts and a state $(\mathbf{a}_1, \ldots, \mathbf{a}_m, \mathsf{st}_{\mathcal{T}}) \leftarrow \mathcal{T}(\mathsf{pk})$ and corresponding ciphertexts $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbf{a}_i)$ using the generic group for each $i \in [m]$. More specifically, $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbf{a}_i)$ samples $r_i \leftarrow_\$ \mathbb{Z}_{p'}$ uniformly at random and for the values r_i and $r_ix_j + \mathbf{a}_i[j]$ for $j \in [n]$ checks whether there already is a mapping in T if not it adds a random label that is uniformly random from the label space $\mathcal{L}$ and different from all existing labels. Now, there are labels $\mathsf{ct}_{i,0}$ and $\mathsf{ct}_{i,j}$ such that the mappings
$$\mathsf{ct}_{i,0} \mapsto r_i, \qquad \text{and} \qquad \mathsf{ct}_{i,j} \mapsto r_ix_j + \mathbf{a}_i[j]$$
   are in the table T.
3. The experiment sends the public key pk and the ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_m$ to the adversary $\mathcal{A}$. When the adversary $\mathcal{A}$ uses its oracle access to the generic group the experiment does the following:

   When the adversary queries the generic group with the two handles ξ_1, ξ_2 the experiment checks whether there are mappings in the table T from ξ_1 and ξ_2 to $\mathbb{Z}_{p'}$ elements Φ_1, Φ_2 respectively. If for $i \in \{1, 2\}$ we have ξ_i does not map to a $\mathbb{Z}_{p'}$ element in T then the experiment samples a new distinct $\mathbb{Z}_{p'}$ element u_i and adds the mapping $\xi_i \mapsto u_i$ into the table T.

Now, ξ_1 and ξ_2 have mappings in T to $\mathbb{Z}_{p'}$ elements Φ_1, Φ_2. Compute $\Phi_3 := \Phi_1 + \Phi_2$. If the table T contains an entry $\xi_3 \mapsto \Phi_3$ for some ξ_3 the experiment forwards ξ_3 to the adversary $\mathcal{A}$. Otherwise, there is no entry for Φ_3 in the table T. The experiment then samples a new distinct label ξ_3 from $\mathcal{L}$ and adds an entry $\xi_3 \mapsto \Phi_3$ to the table T and forwards ξ_3 to the adversary $\mathcal{A}$.

The adversary finally outputs a ciphertext cct, a state $\mathsf{st}_{\mathcal{A}}$ and the trace of its GGM queries tr.

4. The experiment sends the trace tr, the public key pk, the input ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_m$, and the output ciphertext cct to the simulator $\mathcal{S}$. The simulator outputs n functions $(f_j)_{j\in[n]}$.
5. Let $(a'_j)_{j\in[n]} \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{cct})$. The experiment outputs $(\mathsf{st}_{\mathcal{T}}, \mathsf{st}_{\mathcal{A}}, a'_1, \ldots, a'_n)$. That is because for Packed ElGamal decryption algorithm Dec never outputs $\perp$.

Hyb_1**:** Same as Hyb_0 but keeps track of generic group queries with formal variables as in the simulator $\mathcal{S}$. More specifically, the experiment behaves in the following way:

1. In the beginning of the security game, the experiment samples distinct labels $g, (h_j)_{j\in[n]} \leftarrow_\$ \mathcal{L}$. For $j \in [n]$ the experiment samples $x_j \leftarrow_\$ \mathbb{Z}_{p'}$. The experiment initializes an empty table T and adds the mappings $g \mapsto 1$ and $h_j \mapsto \hat{x}_j$ where $\hat{x}_j$ are formal variables. It sets the public key $\mathsf{pk} := (h_j)_{j\in[n]}$ and the secret key $\mathsf{sk} := (x_j)_{j\in[n]}$.
2. Then the experiment samples the plaintexts and a state $(\mathbf{a}_1, \ldots, \mathbf{a}_m, \mathsf{st}_{\mathcal{T}}) \leftarrow \mathcal{T}(\mathsf{pk})$ and corresponding ciphertexts $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbf{a}_i)$ using the generic group for each $i \in [m]$. More specifically, $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbf{a}_i)$ samples $r_i \leftarrow_\$ \mathbb{Z}_{p'}$ uniformly at random and distinct labels $\mathsf{ct}_{i,0}$ and $(\mathsf{ct}_{i,j})_{j\in[n]}$ that are uniformly random under the condition that they are not yet in T. The experiment adds the mappings
$$\mathsf{ct}_{i,0} \mapsto \hat{r}_i, \qquad\qquad \text{and} \qquad\qquad \mathsf{ct}_{i,j} \mapsto \hat{r}_i\hat{x}_j + \hat{\mathbf{a}}_i[j]$$
where $j \in [n]$ to the table T.
3. The experiment sends the public key pk and the ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_m$. When the adversary $\mathcal{A}$ also uses it oracle access to the generic group.

   If the adversary queries the generic group with the two handles ξ_1, ξ_2 the experiment checks whether there are mappings in the table T from ξ_1 and ξ_2 to polynomials over formal variables Φ_1, Φ_2 respectively. If for $i \in \{1, 2\}$ we have ξ_i does not map to a polynomial over formal variables in T and the existing formal variables are $(\hat{u}_j)_{j\in[\ell]}$ then the experiment generates a new formal variable $\hat{u}_{\ell+1}$ and adds the mapping $\xi_i \mapsto \hat{u}_{\ell+1}$ into the table T. Further, the experiment samples a new distinct $\mathbb{Z}_{p'}$ element $u_{\ell+1}$.

   Now, ξ_1 and ξ_2 have mappings in T to polynomials over formal variables Φ_1, Φ_2. Compute $\Phi_3 := \Phi_1 + \Phi_2$. If the table T contains an entry $\xi_3 \mapsto \Phi_3$ for some ξ_3 the experiment forwards ξ_3 to the adversary $\mathcal{A}$. Otherwise, there is no entry for Φ_3 in the table T. The experiment then samples a new distinct label ξ_3 from $\mathcal{L}$ and adds an entry $\xi_3 \mapsto \Phi_3$ to the table T and forwards ξ_3 to the adversary $\mathcal{A}$.

   The adversary finally outputs a ciphertext cct, a state $\mathsf{st}_{\mathcal{A}}$ and the trace of its GGM queries tr.
4. Then the experiment instantiates all the formal variables $(\hat{x}_j)_{j\in[n]}$, $(\hat{r}_i)_{i\in[m]}$, $(\hat{\mathbf{a}}_i[j])$, and $(\hat{u}_i)_{i\in[\ell]}$ by their corresponding $\mathbb{Z}_{p'}$ values $(x_j)_{j\in[n]}$, $(r_i)_{i\in[m]}$, $(\mathbf{a}_i[j])$, and $(u_i)_{i\in[\ell]}$. Now T is a table that maps from labels to $\mathbb{Z}_{p'}$ elements. Continue as in Hyb_0.

Hyb_2**:** Same as Hyb_1 except that checking the well-formedness of the adversary's output $\mathsf{cct} := (\mathsf{cct}_0, (e'_j)_{j\in[n]})$ using the formal variables instead of the $\mathbb{Z}_{p'}$ values. Instead of the final step of Hyb_1 it does the following:

1. At the end of the simulation of the adversary $\mathcal{A}$, it, among other things, outputs a ciphertext $\mathsf{cct} := (\mathsf{cct}_0, (e'_j)_{j\in[n]})$. The experiment checks whether T contains mappings $\mathsf{cct}_0 \mapsto \Phi'_0$, where Φ'_0 is a polynomial over formal variables equivalent to $\sum_{i\in[m]} \alpha_i \hat{r}_i + \beta$ for $\alpha_1, \ldots, \alpha_m, \beta \in \mathbb{Z}_{p'}$. If this is not the case, for $j \in [n]$ compute the experiment samples unused labels v_j outputs the functions:
   - $f_j(\mathbf{a}_1[j], \ldots, \mathbf{a}_m[j])$: Output $(\mathsf{DDL}_g(v_j) - e_j) \mod q$.
2. Then the experiment instantiates all the formal variables $(\hat{x}_j)_{j\in[n]}$, $(\hat{r}_i)_{i\in[m]}$, $(\hat{\mathbf{a}}_i[j])$, and $(\hat{u}_i)_{i\in[\ell]}$ by their corresponding $\mathbb{Z}_{p'}$ values $(x_j)_{j\in[n]}$, $(r_i)_{i\in[m]}$, $(\mathbf{a}_i[j])$, and $(u_i)_{i\in[\ell]}$. Now T is a table that maps from labels to $\mathbb{Z}_{p'}$ elements.
3. The experiment outputs

$$(\mathsf{st}_{\mathcal{T}}, \mathsf{st}_{\mathcal{A}}, f_j(\mathbf{a}_1[1], \ldots, \mathbf{a}_m[1]), \ldots, f_j(\mathbf{a}_1[n], \ldots, \mathbf{a}_m[n])) \ . \tag{1}$$

We argue Hyb_2 is identically distributed to the ideal distribution. This follows from the decryption algorithm $\mathsf{Dec}(\mathsf{sk}, \mathsf{cct} := (\mathsf{cct}_0, (e_j)_{j\in[n]}))$ outputting $\mathsf{DDL}(\mathsf{cct}_j) - e_j$ with $j \in [n]$ and cct_j being the label for the value $\Phi_0 \cdot x_j$, which exactly matches the simulator's output functions after instantiating $(\hat{x}_j)_{j\in[n]}$, $(\hat{r}_i)_{i\in[m]}$, $(\hat{\mathbf{a}}_i[j])_{i\in[m],j\in[n]}$, and $(\hat{u}_i)_{i\in[\ell]}$ by their corresponding values $(x_j)_{j\in[n]}$, $(r_i)_{i\in[m]}$, $(\mathbf{a}_i[j])_{i\in[m],j\in[n]}$, and $(u_i)_{i\in[\ell]}$. That is because after instantiation

$$\sum_i \alpha_i(\hat{r}_i\hat{x}_j + \hat{\mathbf{a}}_i[j]) + \beta\hat{x}_j - \sum_i \alpha_i \mathbf{a}_i[j]$$

becomes

$$\sum_i \alpha_i(r_i x_j + \mathbf{a}_i[j]) + \beta x_j - \sum_i \alpha_i \mathbf{a}_i[j] = \sum_i \alpha_i r_i x_j + \beta x_j - \sum_i = \Phi_0 \cdot x_j.$$

Therefore, $f_j(\mathbf{a}_1[j], \ldots, \mathbf{a}_m[j]) = a'_j$ and the distributions are the same.

Finally, we argue statistical distance of the hybrids for polynomial generic group queries. Statistical distance between Hyb_0 and Hyb_1 is established in Claim 5.4 and the statistical distance between Hyb_1 and Hyb_2 in Claim 5.5. The sum of these distances is $< 4t^2/p'$ for $t > 4t_{\mathsf{Dec}} + 1$, where t_{Dec} is the number of queries the decryption algorithm does. $\square$

Claim 5.4. For any adversary $\mathcal{A}$ making t many queries to the generic group oracle $\mathsf{Hyb}_0(\mathcal{A})$ has a statistical distance of $3t(t+1)/p'$ from $\mathsf{Hyb}_1(\mathcal{A})$.

Proof. We show that the view of the adversary $\mathcal{A}$ is statistically close in the two hybrid if it only makes polynomially many queries to the generic group oracle. We argue that the public key pk, the ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_m$ and adversary's queries to the generic group are statistically close in Hyb_0 and Hyb_1. We argue via a hybrid argument over each query to the generic group. We start with $\mathsf{Hyb}_{0,0}$ which is identically distributed to Hyb_0 and with $\mathsf{Hyb}_{0,s}$ the first s computed elements are as in Hyb_1 and all the rest are still as in Hyb_0. We then show that $\mathsf{Hyb}_{0,s-1}$ and $\mathsf{Hyb}_{0,s}$ are statistically close. It suffices to consider the i-th computed element:

- The hybrids only differ in behavior if the label ξ given to the adversary maps to a polynomial over formal variables Φ that has a non-zero linear term in $(\hat{r}_i)_{i\in[m]}$, $(\hat{r}_i\hat{x}_j)_{i\in[m],j\in[n]}$, or $(\hat{u}_i)_{i\in[\ell]}$ monomials while $\Phi((x_j)_{j\in[n]}, (r_i)_{i\in[m]}, (\mathbf{a}_i[j])_{i\in[m],j\in[n]}, (u_i)_{i\in[\ell]})$ is already in the table T. This condition is equivalent to the condition that there exists a constant $c \in \mathbb{Z}_{p'}$ in the table T such that $\Phi((\hat{x}_j)_{j\in[n]}, (\hat{r}_i)_{i\in[m]}, (\hat{\mathbf{a}}_i[j])_{i\in[m],j\in[n]}, (\hat{u}_i)_{i\in[\ell]}) - c \not\equiv 0$, but $\Phi((x_j)_{j\in[n]}, (r_i)_{i\in[m]}, (\mathbf{a}_i[j])_{i\in[m],j\in[n]}, (u_i)_{i\in[\ell]}) - c = 0$. We show this happens with negligible probability. In both, $\mathsf{Hyb}_{0,s-1}$ and $\mathsf{Hyb}_{0,s}$ the first $s-1$ elements are handled as in Hyb_1. Therefore, these elements are independent of $(x_j)_{j\in[n]}$, $(r_i)_{i\in[m]}$, $(\mathbf{a}_i[j])_{i\in[m],j\in[n]}$, $(u_i)_{i\in[\ell]}$. Fix an arbitrary constant $c \in \mathbb{Z}_{p'}$ from the table T. We define a new polynomial

$$\begin{aligned}&\Psi_c((\hat{x}_j)_{j\in[n]}, (\hat{r}_i)_{i\in[m]}, (\hat{u}_i)_{i\in[\ell]})\\ =&\Phi((\hat{x}_j)_{j\in[n]}, (\hat{r}_i)_{i\in[m]}, (\mathbf{a}_i[j])_{i\in[m],j\in[n]}, (\hat{u}_i)_{i\in[\ell]}) - c.\end{aligned}$$

Notice, that we instantiated $(\hat{\mathbf{a}}_i[j])_{i\in[m],j\in[n]}$ by $(\mathbf{a}_i[j])_{i\in[m],j\in[n]}$. In the polynomial $\Phi((\hat{x}_j)_{j\in[n]}, (\hat{r}_i)_{i\in[m]}, (\hat{\mathbf{a}}_i[j])_{i\in[m],j\in[n]}, (\hat{u}_i)_{i\in[\ell]})$ we have monomial $\hat{\mathbf{a}}_i[j]$ always with the same arity as the corresponding monomial $\hat{r}_i\hat{x}_j$ for $i \in [m], j \in \{0,1\}$ because the challenger initially only gives out labels for polynomials of the form $\hat{r}_i\hat{x}_j + \hat{\mathbf{a}}_i[j]$ and the oracles only allow the adversary to learn linear combinations. Therefore, if the polynomial Ψ_c is the zero polynomial then so is $\Phi((\hat{x}_j)_{j\in[n]}, (\hat{r}_i)_{i\in[m]}(\hat{\mathbf{a}}_i[j])_{i\in[m],j\in[n]}, (\hat{u}_i)_{i\in[\ell]}) - c$.

Because Ψ_c is a polynomial of total degree 2 we derive by polynomial identity lemma that

$$\Pr[\Psi_c((\hat{x}_j)_{j\in[n]}, (\hat{r}_i)_{i\in[m]}, (\hat{u}_i)_{i\in[\ell]}) = 0] \leq 2/p'.$$

By union bound we get that

$$\Pr[\exists c \in \mathbb{Z}_{p'} \cap \mathsf{T} : \Psi_c((\hat{x}_j)_{j\in[n]}, (\hat{r}_i)_{i\in[m]}, (\hat{u}_i)_{i\in[\ell]}) = 0] \leq 2|\mathsf{T}|/p'.$$

Thus, with probability $2|\mathsf{T}|/p'$ for Φ degree ≥ 1 we have $\Phi((x_j)_{j\in[n]}, (r_i)_{i\in[m]}, (\mathbf{a}_i[j])_{i\in[m],j\in[n]})$ is not in the table T. Therefore, $\mathsf{Hyb}_{0,s-1}$ and $\mathsf{Hyb}_{0,s}$ have the same behavior with probability $1 - 2|\mathsf{T}|/p'$.

If the number of queries to the generic group oracle is ℓ then $\mathsf{Hyb}_{0,0}$ is identically distributed to Hyb_0 and $\mathsf{Hyb}_{0,\ell}$ is identically distributed to Hyb_1. Because every query introduces at most 3 entries into T we get the statistical distance between Hyb_0 and Hyb_1 is at most $\sum_{i\in[\ell]} 2(3i)/p' = 3\ell(\ell+1)/p'$.

$\square$

Claim 5.5. For any adversary $\mathcal{A}$ making $t_{\mathcal{A}}$ many queries and Dec making t_{Dec} queries to the generic group oracle $\mathsf{Hyb}_1(\mathcal{A})$ has a statistical distance of $3t_{\mathsf{Dec}}(t_{\mathcal{A}} + (t_{\mathsf{Dec}}+1)/2)/p'$ from $\mathsf{Hyb}_2(\mathcal{A})$.

Proof. The view of the adversary in Hyb_1 and Hyb_2 are identical, therefore, all that is left to prove is that decryption of the output ciphertext is statistically close. Notice that the adversary's view is independent of $(x_j)_{j\in[n]}$, $(r_i)_{i\in[m]}$, $(\mathbf{a}_i[j])_{i\in[m],j\in[n]}$, and $(u_i)_{i\in[\ell]}$, therefore we can treat them as being sampled after the adversary outputs the ciphertext. We analyze the following three scenarios:

- If the label in the output ciphertext cct_0 is not in the table T: Without loss of generality this does not happen because one can always introduce a new formal variable $\hat{u}_{\ell+1}$, where $(\hat{u}_i)_{i\in[\ell]}$ are the variables that represent unknown elements, and add $\mathsf{cct}_0 \mapsto \hat{u}_{\ell+1}$ to the table T and this case reduces to the next one.

- The table T contains a mapping from the label of the output ciphertext $\mathsf{cct} := (\mathsf{cct}_0, (e'_j)_{j\in[n]})$ to a polynomial Φ'_0 such that $\Phi'_0 \not\equiv \sum_{i\in[m]} \alpha_i \hat{r}_i + \beta$ for $\alpha_1, \ldots \alpha_m, \beta \in \mathbb{Z}_{p'}$:

  Now, we have that Φ'_0 contains one of the following monomials: $\hat{r}_i\hat{x}_j$, $\hat{x}_j$, or $(\hat{u}_i)_{i\in[\ell]}$ for $i \in [m]$ and $j \in [n]$. Therefore, the adversary interacted with the group oracle involving the label cct_ℓ (either used is as input or received it as output) for $\Phi'_0 \cdot \hat{x}_\ell$ with $t \in [n]$ as it can only linearly combine 1, $(\hat{x}_j)_{j\in[n]}$, $(r_i\hat{x}_j + \hat{\mathbf{a}}_i[j])_{i\in[m],j\in[n]}$, and $(\hat{u}_i)_{i\in[\ell]}$. Therefore, Hyb_1 and Hyb_2 only differ if in Hyb_1 the $\mathsf{DDL}(\mathsf{cct}_t)$ algorithm queries an entry that has already been assigned to a value. We argue via a sequence of hybrids that the probability of this happening is negligible. To prove this we use the property that $\mathsf{DDL}_g(\mathsf{cct}_t)$ only has access to g (the label for 1) and its input label cct_t. We start with $\mathsf{Hyb}_{1,0}$ which is identically distributed to Hyb_1 and then in $\mathsf{Hyb}_{1,s}$ the first i queries are handled with formal variables. We now show that $\mathsf{Hyb}_{1,s-1}$ and $\mathsf{Hyb}_{1,s}$ are statistically close. It suffices to consider the s-th computed element:

  - The hybrids only differ in behavior if for some $j \in [n]$ a label ξ that $\mathsf{DDL}(\mathsf{cct}_0^{x_j})$ received from the generic group orale maps to a degree ≥ 1 polynomial Φ over formal variables while $\Phi((x_j)_{j\in[n]}, (r_i)_{i\in[m]}, (\mathbf{a}_i[j])_{i\in[m],j\in[n]}, (u_i)_{i\in[\ell]})$ is already in the table T. This condition is equivalent to the condition that there exists a constant $c \in \mathbb{Z}'_p$ in the table T such that $\Phi((\hat{x}_j)_{j\in[n]}, (\hat{r}_i)_{i\in[m]}, (\hat{\mathbf{a}}_i[j])_{i\in[m],j\in[n]}, (\hat{u}_i)_{i\in[\ell]}) - c \not\equiv 0$, but $\Phi((x_j)_{j\in[n]}, (r_i)_{i\in[m]}, (\mathbf{a}_i[j])_{i\in[m],j\in[n]}, (u_i)_{i\in[\ell]}) - c = 0$.

We show this happens with negligible probability. In both, $\mathsf{Hyb}_{1,s-1}$ and $\mathsf{Hyb}_{1,s}$ the first $s-1$ elements are handled with formal variables.

Therefore, $\mathsf{DDL}(\mathsf{cct}_\ell)$ has only the adversary interacted with the group oracle involving labels for linear combinations of 1, $\Phi_0' \cdot \hat{x}_t$, and formal variables $(\hat{u}_{i+\ell})_{i\in[s-1]}$. So, the s-th query will be a linear combination of 1, $\Phi_0' \cdot \hat{x}_t$, and formal variables $(\hat{u}_{i+\ell})_{i\in[s]}$ (it might have introduced a new formal variable). More precisely, this means that the query will output a label that maps to $\alpha + \beta \cdot \Phi_0' \cdot \hat{x}_t + \sum_{i\in[s]} \gamma_i \hat{u}_{i+\ell}$ for some $\alpha, \beta, \gamma_i \in \mathbb{Z}_p'$. If $\beta = \gamma_1 = ... = \gamma_s = 0$ then the formal polynomial is not of degree ≥ 1 meaning $\mathsf{Hyb}_{1,s-1}$ and $\mathsf{Hyb}_{1,s}$ are identically distributed.

Further, as they only depend on formal variables, these elements are independent of $(x_j)_{j\in[n]}$, $(r_i)_{i\in[m]}$, $(\mathbf{a}_i[j])_{i\in[m],j\in[n]}$, $(u_i)_{i\in[\ell+s]}$. Fix an arbitrary constant $c \in \mathbb{Z}_p'$ from the table T. We define a new polynomial

$$\begin{aligned}&\Psi_{c,t}((\hat{x}_j)_{j\in[n]}, (\hat{r}_i)_{i\in[m]}, (\hat{u}_i)_{i\in[\ell+s]})\\ =&\alpha + \beta \cdot \Phi_0'((\hat{x}_j)_{j\in[n]}, (\hat{r}_i)_{i\in[m]}, (\hat{u}_i)_{i\in[\ell+s]}, (\mathbf{a}_i[j])_{i\in[m],j\in[n]}) \cdot \hat{x}_t - c.\end{aligned}$$

Notice, that we instantiated $(\hat{\mathbf{a}}_i[j])_{i\in[m],j\in[n]}$ by $(\mathbf{a}_i[j])_{i\in[m],j\in[n]}$. The polynomial $\Phi((\hat{x}_j)_{j\in[n]}$, $(\hat{r}_1)_{i\in[m]}$, $(\hat{\mathbf{a}}_i[j])_{i\in[m],j\in[n]}$, $(\hat{u}_i)_{i\in[\ell]})$ must at least contain one of the following monomials:

$$(\hat{r}_i\hat{x}_j\hat{x}_t)_{i\in[m],j,t\in[n]}, \qquad (\hat{x}_j\hat{x}_t)_{j,t\in[n]}, \qquad \text{or} \qquad (\hat{u}_i\hat{x}_t)_{i\in[\ell+s],t\in[n]}.$$

Therefore, if the polynomial $\Psi_{c,t}$ is the zero polynomial then so is $\Phi((\hat{x}_j)_{j\in[n]}, (\hat{r}_1)_{i\in[m]}, (\hat{\mathbf{a}}_i[j])_{i\in[m],j\in[n]}, (\hat{u}_i)_{i\in[\ell+s]}) - c$.

Because Φ_0' is an comes from an output of the adversary and the adversary only knows polynomials of degree ≤ 2 and can only compute linear combinations we have Φ_0' is also of degree ≤ 2. It follows that $\Psi_{c,t}$ is ≤ 3 degree polynomial. Then, we derive by polynomial identity lemma that

$$\Pr[\Psi_{c,t}((x_j)_{j\in[n]}, (r_i)_{i\in[m]}, (u_i)_{i\in[\ell]}) = 0] \leq 3/p'.$$

By union bound we get that

$$\Pr[\exists c \in \mathbb{Z}_{p'} \cap \mathsf{T} : \Psi_{c,t}((x_j)_{j\in[n]}, (r_i)_{i\in[m]}, (u_i)_{i\in[\ell]}) = 0] \leq 3|\mathsf{T}|/p'.$$

Thus, with probability $3|\mathsf{T}|/p'$ for Φ with degree ≥ 1 we have $\Phi((x_j)_{j\in[n]}, (r_i)_{i\in[m]}, (a_i[j])_{i\in[m],j\in[n]})$ is not in the table T. Therefore, $\mathsf{Hyb}_{0,s-1}$ and $\mathsf{Hyb}_{0,s}$ have the same behavior with probability $1 - 3|\mathsf{T}|/p'$.

We have $\mathsf{Hyb}_{1,0}$ is identically distributed to Hyb_1. Let $\ell_{\mathcal{A}}$ be the number of generic group oracle calls the adversary made and ℓ_{Dec} be the number of generic group oracle call Dec makes.

Further, the statistical distance between $\mathsf{Hyb}_{1,0}$ and $\mathsf{Hyb}_{1,\ell_{\mathsf{Dec}}}$ is $\sum_{i\in[\ell_{\mathsf{Dec}}]} 3(\ell_{\mathcal{A}} + i)/p' = 3\ell_{\mathsf{Dec}}(\ell_{\mathcal{A}} + (\ell_{\mathsf{Dec}} + 1)/2)/p'$ because each query can introduce at most 3 terms.

Then Hyb_2 is identically distributed to $\mathsf{Hyb}_{1,\ell_{\mathsf{Dec}}}$ because as argued above the adversary does not have access to the formal polynomial $\Phi_0' \cdot \hat{x}_t$, therefore, from its perspective the corresponding label could also be chosen uniformly at random from the labels the adversary has not seen yet.

- The table T contains a mapping from each label of the output ciphertext $\mathsf{cct} := (\mathsf{cct}_0, (e_j')_{j\in[n]})$ to a polynomial Φ_0' such that $\Phi_0' \equiv \sum_{i\in[m]} \alpha_i \hat{r}_i + \beta$ for $\alpha_1, \ldots \alpha_m, \beta \in \mathbb{Z}_{p'}$:

  Hyb_1 and Hyb_2 behave identically in that case.

□

5.3 Bound-limited homomorphism of Packed ElGamal with hash check

In this section, we prove that Packed ElGamal with hash check has bound limited homomorphism.

Lemma 5.6. *Packed ElGamal with hash check for has $8Bn$-bound-limited homomorphism with distinguishability error $\varepsilon_{\mathsf{bnd}}(\lambda, t, \chi, m) = 4t^2/p' + 2t^2/2^\chi \le 6t^2/2^\lambda$ for $t > 4t_{\mathsf{Dec}} + 1$, where t_{Dec} is the number of queries the decryption algorithm does.*

Proof. For any adversary $\mathcal{A}$ and plaintext generator $\mathcal{T}$ in the bound-limited homomorphism experiment we define an extractor Ext:

1. In the beginning of the security game, the extractor receives the trace tr, the public key $\mathsf{pk} := (h_1, h_2)$, and $\mathsf{ct}_i := (\mathsf{ct}_{i,0}, (\mathsf{ct}_{i,j})_{j\in[n]})$ for $i \in [m]$. The extractor initializes an empty table T and for $i \in [m]$ and $j \in [n]$ adds the expressions

$$g \mapsto 1, \qquad h_j \mapsto \hat{x}_j, \qquad \mathsf{ct}_{i,0} \mapsto \hat{r}_i, \qquad \text{and} \qquad \mathsf{ct}_{i,j} \mapsto \hat{r}_i\hat{x}_j + \hat{\mathbf{a}}_i[j]$$

   to the table where $\hat{x}_j$, $\hat{r}_i$, and $\hat{a}_i[j]$ are formal variables representing secret key, randomness and messages respectively for $i \in [m]$.

2. The simulator goes through the trace tr first to last entry and does the following for each entry:

   Each entry has two input labels handles ξ_1 and ξ_2, the simulator checks whether there are mappings T from ξ_1 and ξ_2 to polynomials over formal variables Φ_1 and Φ_2 in the table, respectively. If ξ_i does not map to a polynomial in the table T and the existing formal variables are $(\hat{u}_j)_{j\in[\ell]}$ then the simulator generates a new formal variable $\hat{u}_{\ell+1}$ and adds the mapping $\xi_i \mapsto \hat{u}_{\ell+1}$ into the table T. Now, ξ_1 and ξ_2 both have mappings in T to polynomials over formal variables Φ_1, Φ_2.

   The simulator computes the polynomial $\Phi_3 := \Phi_1 + \Phi_2$. The simulator looks at the output label of the trace entry ξ_3. If the table T contains an entry $\xi_3 \mapsto \Phi_3'$ for some polynomial Φ_3' then the simulator outputs $\perp$ if $\Phi_3' \not\equiv \Phi_3$.

3. The simulator also has the adversary's output $\mathsf{cct} := (\mathsf{cct}_0, (e_j)_{j\in[n]}, k)$. The simulator checks whether T contains mappings $\mathsf{cct}_0 \mapsto \Phi_0'$, where Φ_0' is a polynomial over formal variables equivalent to $\sum_{i\in[m]} \alpha_i\hat{r}_i + \beta$ for $\alpha_1, \ldots, \alpha_m, \beta \in \mathbb{Z}_{p'}$. If this is not the case, the simulator outputs $\perp$.

4. For each $j \in [n]$ the simulator computes labels cct_j corresponding to $\beta\hat{x}_j + \sum_{i\in[m]} \alpha_i\hat{r}_i\hat{x}_j$.

5. For each $j \in [n]$ the simulator computes $v_j \leftarrow \mathsf{DDL}(\mathsf{cct}_j)$.

6. If the extractor has reached this point it outputs $(\Pi = (\alpha_1, \ldots, \alpha_m), \mathbf{b} = (v_j - e_j \mod p)_{j\in[n]})$.

We show the simulated linear function outputs the correct value if the output ciphertext decrypts successfully with all but negligible probability. We prove this via hybrid argument.

Hyb_0**:** It is the same as the real distribution in Definition 5.2. In more detail:

1. In the beginning of the security game, for each $j \in [n]$ the experiment samples $x_j \leftarrow_\$ \mathbb{Z}_{p'}$. The experiment initializes an empty table T and adds the mapping $g \mapsto 1$. For $j \in [n]$ the experiment checks whether there already exists a mapping from to x_j. If not the experiment samples a new distinct label h_j and adds $h_j \mapsto x_j$ to the table T. It sets the public key $\mathsf{pk} := (h_j)_{j\in[n]}$ and the secret key $\mathsf{sk} := (x_j)_{j\in[n]}$.
2. Then the experiment samples the plaintexts and a state $(\mathbf{a}_1, \ldots, \mathbf{a}_m, \mathsf{st}_{\mathcal{T}}) \leftarrow \mathcal{T}(\mathsf{pk})$ and corresponding ciphertexts $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbf{a}_i)$ using the generic group for each $i \in [m]$. More specifically, $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbf{a}_i)$ samples $r_i \leftarrow_\$ \mathbb{Z}_{p'}$ uniformly at random and for the values r_i and $r_ix_j + \mathbf{a}_i[j]$ for $j \in [n]$ checks whether there already is a mapping in T if not it adds a random label that is uniformly random from the label space $\mathcal{L}$ and different from all existing labels. Now, there are labels $\mathsf{ct}_{i,0}$ and $\mathsf{ct}_{i,j}$ such that the mappings

$$\mathsf{ct}_{i,0} \mapsto r_i, \qquad \text{and} \qquad \mathsf{ct}_{i,j} \mapsto r_ix_j + \mathbf{a}_i[j]$$

   are in the table T.

3. The experiment sends the public key pk and the ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_m$ to the adversary $\mathcal{A}$. When the adversary $\mathcal{A}$ uses its oracle access to the generic group the experiment does the following:

   When the adversary queries the generic group with the two handles ξ_1, ξ_2 the experiment checks whether there are mappings in the table T from ξ_1 and ξ_2 to $\mathbb{Z}_{p'}$ elements Φ_1, Φ_2 respectively. If for $i \in \{1, 2\}$ we have ξ_i does not map to a $\mathbb{Z}_{p'}$ element in T then the experiment samples a new distinct $\mathbb{Z}_{p'}$ element u_i and adds the mapping $\xi_i \mapsto u_i$ into the table T.

   Now, ξ_1 and ξ_2 have mappings in T to $\mathbb{Z}_{p'}$ elements Φ_1, Φ_2. Compute $\Phi_3 := \Phi_1 + \Phi_2$. If the table T contains an entry $\xi_3 \mapsto \Phi_3$ for some ξ_3 the experiment forwards ξ_3 to the adversary $\mathcal{A}$. Otherwise, there is no entry for Φ_3 in the table T. The experiment then samples a new distinct label ξ_3 from $\mathcal{L}$ and adds an entry $\xi_3 \mapsto \Phi_3$ to the table T and forwards ξ_3 to the adversary $\mathcal{A}$.

   The adversary finally outputs a ciphertext cct, a state $\mathsf{st}_{\mathcal{A}}$ and the trace of its GGM queries tr.

4. The experiment sends the trace tr, the public key pk, the input ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_m$, and the output ciphertext cct to the simulator $\mathcal{S}$. The simulator outputs Π and $\mathbf{b}$.

5. If $\mathsf{Dec}(\mathsf{sk}, \mathsf{cct}) = \bot$ the experiment outputs $\bot$. More specifically $\mathsf{Dec}(\mathsf{sk}, \mathsf{cct} := (\mathsf{cct}_0, (e'_j)_{j\in[n]}, k'))$ checks if
$$k = \mathsf{H}(c_0, (c_0^{x_j} \cdot g^{\mathsf{DDL}(c_0^{x_j})})_{j\in[n]}).$$
   If this is not the case the experiment outputs $\bot$.

6. Let $(a'_j)_{j\in[n]} \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{cct})$. The experiment outputs $(\mathsf{st}_{\mathcal{T}}, \mathsf{st}_{\mathcal{A}}, a'_1, \ldots, a'_n)$.

Hyb_1**:** Same as Hyb_0 but keeps track of generic group queries with formal variables as in the simulator $\mathcal{S}$. More specifically, the experiment behaves in the following way:

1. In the beginning of the security game, the experiment samples distinct labels $g, (h_j)_{j\in[n]} \leftarrow_\$ \mathcal{L}$. For $j \in [n]$ the experiment samples $x_j \leftarrow_\$ \mathbb{Z}_{p'}$. The experiment initializes an empty table T and adds the mappings $g \mapsto 1$ and $h_j \mapsto \hat{x}_j$ where $\hat{x}_j$ are formal variables. It sets the public key $\mathsf{pk} := (h_j)_{j\in[n]}$ and the secret key $\mathsf{sk} := (x_j)_{j\in[n]}$.

2. Then the experiment samples the plaintexts and a state $(\mathbf{a}_1, \ldots, \mathbf{a}_m, \mathsf{st}_{\mathcal{T}}) \leftarrow \mathcal{T}(\mathsf{pk})$ and corresponding ciphertexts $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbf{a}_i)$ using the generic group for each $i \in [m]$. More specifically, $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbf{a}_i)$ samples $r_i \leftarrow_\$ \mathbb{Z}_{p'}$ uniformly at random and distinct labels $\mathsf{ct}_{i,0}$ and $(\mathsf{ct}_{i,j})_{j\in[n]}$ that are uniformly random under the condition that they are not yet in T. The experiment adds the mappings
$$\mathsf{ct}_{i,0} \mapsto \hat{r}_i, \qquad \text{and} \qquad \mathsf{ct}_{i,j} \mapsto \hat{r}_i\hat{x}_j + \hat{\mathbf{a}}_i[j]$$
   where $j \in [n]$ to the table T.

3. The experiment sends the public key pk and the ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_m$. When the adversary $\mathcal{A}$ also uses it oracle access to the generic group.

   If the adversary queries the generic group with the two handles ξ_1, ξ_2 the experiment checks whether there are mappings in the table T from ξ_1 and ξ_2 to polynomials over formal variables Φ_1, Φ_2 respectively. If for $i \in \{1, 2\}$ we have ξ_i does not map to a polynomial over formal variables in T and the existing formal variables are $(\hat{u}_j)_{j\in[\ell]}$ then the experiment generates a new formal variable $\hat{u}_{\ell+1}$ and adds the mapping $\xi_i \mapsto \hat{u}_{\ell+1}$ into the table T. Further, the experiment samples a new distinct $\mathbb{Z}_{p'}$ element $u_{\ell+1}$.

   Now, ξ_1 and ξ_2 have mappings in T to polynomials over formal variables Φ_1, Φ_2. Compute $\Phi_3 := \Phi_1 + \Phi_2$. If the table T contains an entry $\xi_3 \mapsto \Phi_3$ for some ξ_3 the experiment forwards ξ_3 to the adversary $\mathcal{A}$. Otherwise, there is no entry for Φ_3 in the table T. The experiment then samples a new distinct label ξ_3 from $\mathcal{L}$ and adds an entry $\xi_3 \mapsto \Phi_3$ to the table T and forwards ξ_3 to the adversary $\mathcal{A}$.

   The adversary finally outputs a ciphertext cct, a state $\mathsf{st}_{\mathcal{A}}$ and the trace of its GGM queries tr.

4. Then the experiment instantiates all the formal variables $(\hat{x}_j)_{j\in[n]}$, $(\hat{r}_i)_{i\in[m]}$, $(\hat{\mathbf{a}}_i[j])$, and $(\hat{u}_i)_{i\in[\ell]}$ by their corresponding $\mathbb{Z}_{p'}$ values $(x_j)_{j\in[n]}$, $(r_i)_{i\in[m]}$, $(\mathbf{a}_i[j])$, and $(u_i)_{i\in[\ell]}$. Now T is a table that maps from labels to $\mathbb{Z}_{p'}$ elements. Continue as in Hyb_0.

Hyb_2**:** Same as Hyb_1 but instead of the last step the experiment does the following:

1. For the ciphertext $\mathsf{cct} := (\mathsf{cct}_0, (e_j)_{j\in[n]}, k)$ if there is no mapping $\mathsf{cct}_0 \mapsto \Phi'_0$ in the table T such that $\Phi'_0 \equiv \sum_{i\in[m]} \alpha_i \hat{r}_i + \beta$ for $\alpha_1, \dots \alpha_m, \beta \in \mathbb{Z}_{p'}$ the experiment samples n uniform unused labels $(\mathsf{cct}_j)_{j\in[n]}$.
2. If $\mathsf{H}(\mathsf{cct}_0, (\mathsf{cct}_j \cdot g^{\mathsf{DDL}(\mathsf{cct}_j)})_{j\in[n]}) \neq k$ the experiment outputs $\perp$.
3. Otherwise, the experiment instantiates all the formal variables $(\hat{x}_j)_{j\in[n]}$, $(\hat{r}_i)_{i\in[m]}$, $(\hat{\mathbf{a}}_i[j])_{i\in[m],j\in[n]}$, and $(\hat{u}_i)_{i\in[\ell]}$ in T by their corresponding values $(x_j)_{j\in[n]}$, $(r_i)_{i\in[m]}$, $(\mathbf{a}_i[j])_{i\in[m],j\in[n]}$, and $(u_i)_{i\in[\ell]}$. Now T is a table that maps from labels to $\mathbb{Z}_{p'}$ elements.
4. The experiment computes $(a'_j)_{j\in[n]} \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{cct})$. The experiment outputs $(\mathsf{st}_{\mathcal{T}}, \mathsf{st}_{\mathcal{A}}, a'_1, \dots, a'_n)$.

Hyb_3**:** Same as Hyb_2 but instead of creating dummy variables if there is no mapping $\mathsf{cct}_0 \mapsto \Phi'_0$, we simply output $\perp$.

1. For the ciphertext $\mathsf{cct} := (\mathsf{cct}_0, (e_j)_{j\in[n]}, k)$ if there is no mapping $\mathsf{cct}_0 \mapsto \Phi'_0$ in the table T such that $\Phi'_0 \equiv \sum_{i\in[m]} \alpha_i \hat{r}_i + \beta$ for $\alpha_1, \dots \alpha_m, \beta \in \mathbb{Z}_{p'}$ the experiment outputs $\perp$.
2. If $\mathsf{H}(\mathsf{cct}_0, (\mathsf{cct}_j \cdot g^{\mathsf{DDL}(\mathsf{cct}_j)})_{j\in[n]}) \neq k$ the experiment outputs $\perp$.
3. Otherwise, the experiment instantiates all the formal variables $(\hat{x}_j)_{j\in[n]}$, $(\hat{r}_i)_{i\in[m]}$, $(\hat{\mathbf{a}}_i[j])_{i\in[m],j\in[n]}$, and $(\hat{u}_i)_{i\in[\ell]}$ in T by their corresponding values $(x_j)_{j\in[n]}$, $(r_i)_{i\in[m]}$, $(\mathbf{a}_i[j])_{i\in[m],j\in[n]}$, and $(u_i)_{i\in[\ell]}$. Now T is a table that maps from labels to $\mathbb{Z}_{p'}$ elements.
4. The experiment computes $(a'_j)_{j\in[n]} \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{cct})$. The experiment outputs $(\mathsf{st}_{\mathcal{T}}, \mathsf{st}_{\mathcal{A}}, a'_1, \dots, a'_n)$.

Hyb_4**:** Same as Hyb_3 but instead of the final step the experiment now does the following:

1. For a $j \in [n]$, with cct_j being the label in T for $\beta\hat{x}_j + \sum_{i\in[m]} \alpha_i(\hat{r}_i\hat{x}_j + \hat{\mathbf{a}}_i[j])$, and cct^*_j being the label for $\beta\hat{x}_j + \sum_{i\in[m]} \alpha_i(\hat{r}_i\hat{x}_j + \hat{\mathbf{a}}_i[j] - \mathbf{a}_i[j])$ if
$$\mathsf{cct}^*_j \cdot g^{\mathsf{DDL}(\mathsf{cct}^*_j)} \neq \mathsf{cct}_j \cdot g^{\mathsf{DDL}(\mathsf{cct}_j)}.$$
the experiment outputs $\perp$.
2. Otherwise, the experiment instantiates all the formal variables $(\hat{x}_j)_{j\in[n]}$, $(\hat{r}_i)_{i\in[m]}$, $(\hat{\mathbf{a}}_i[j])_{i\in[m],j\in[n]}$, and $(\hat{u}_i)_{i\in[\ell]}$ in T by their corresponding values $(x_j)_{j\in[n]}$, $(r_i)_{i\in[m]}$, $(\mathbf{a}_i[j])_{i\in[m],j\in[n]}$, and $(u_i)_{i\in[\ell]}$. Now T is a table that maps from labels to $\mathbb{Z}_{p'}$ elements.
3. The experiment computes $(a'_j)_{j\in[n]} \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{cct})$. The experiment outputs $(\mathsf{st}_{\mathcal{T}}, \mathsf{st}_{\mathcal{A}}, a'_1, \dots, a'_n)$.

We argue that in Hyb_4 the adversaries winning probability is 0. First, note that if for all $j \in [n]$ we have $\mathsf{cct}^*_j \cdot g^{\mathsf{DDL}(\mathsf{cct}^*_j)} = \mathsf{cct}_j \cdot g^{\mathsf{DDL}(\mathsf{cct}_j)}$ then $\sum_{i\in[m]} \alpha_i \mathbf{a}_i[j] \in [-8Bn, 8Bn]$ by contrapositon of Lemma 3.14 because DDL does $8Bn$ queries.

Then, we argue that decryption outputs exactly what the extractor outputs. cct^*_j is a label for $\beta\hat{x}_j + \sum_{i\in[m]} \alpha_i(\hat{r}_i\hat{x}_j + \hat{\mathbf{a}}_i[j] - \mathbf{a}_i[j])$. After instantiating $(\hat{r}_i)_{i\in[m]}$, $\hat{x}_j$, and $(\hat{\mathbf{a}}_i[j])_{i\in[m]}$ by their values $(r_i)_{i\in[m]}$, x_j, and $(\mathbf{a}_i[j])_{i\in[m]}$ this exactly corresponds to $\sum_{i\in[m]} \alpha_i r_i x_j + \beta x_j$. Therefore, Dec computes $\mathsf{DDL}(\mathsf{cct}^*_j)$ and outputs
$$(\mathsf{DDL}(\mathsf{cct}^*_j) - e_j) \mod p = (\mathsf{DDL}(\mathsf{cct}_j) + \sum_{i\in[m]} \alpha_i \mathbf{a}_i[j] - e_j) \mod p.$$

This is exactly what the experiment outputs in the ideal world because $\mathsf{DDL}(\mathsf{cct}^*_j)$ and e_j are much smaller than the cryptographic group.

Finally, we argue statistical distance of the hybrids for polynomial generic group queries. Statistical distance between Hyb_0 and Hyb_1 is established using the exact same arguments as Claim 5.4 and the statistical distance between Hyb_1 and Hyb_2 is established using the exact same arguments as Claim 5.5. Statistical distance between Hyb_2 and Hyb_4 is established in Claims 5.7 and 5.8. The sum of these distances is $< (t+1)^2/p'$ for $t > 4t_{\mathsf{Dec}} + 1$, where t_{Dec} is the number of queries the decryption algorithm does. □

Claim 5.7. For any adversary $\mathcal{A}$ we have $\mathsf{Hyb}_2(\mathcal{A})$ and $\mathsf{Hyb}_3(\mathcal{A})$ are at statistical distance $2^{-\chi}$.

Proof. Follows from H being a universal hash function with a codomain of $\{0,1\}^{\chi}$. □

Claim 5.8. For any adversary $\mathcal{A}$ with t queries to the ROM we have $\mathsf{Hyb}_3(\mathcal{A})$ are at statistical distance of $t^2/2^{\chi}$ to $\mathsf{Hyb}_4(\mathcal{A})$.

Proof. We prove that checking equality between

$$(\mathsf{cct}_j \cdot g^{\mathsf{DDL}(\mathsf{cct}_j)})_{j\in[n]} \qquad \text{and} \qquad (\mathsf{cct}^*_j \cdot g^{\mathsf{DDL}(\mathsf{cct}^*_j)})_{j\in[n]}$$

is statistically close close to checking the equality of their hashes. This follows from collision resistance of H. More precisely, if $(\mathsf{cct}_j \cdot g^{\mathsf{DDL}(\mathsf{cct}_j)})_{j\in[n]} \neq (\mathsf{cct}^*_j \cdot g^{\mathsf{DDL}(\mathsf{cct}^*_j)})_{j\in[n]}$ but $\mathsf{H}((\mathsf{cct}_j \cdot g^{\mathsf{DDL}(\mathsf{cct}_j)})_{j\in[n]}) = \mathsf{H}((\mathsf{cct}^*_j \cdot g^{\mathsf{DDL}(\mathsf{cct}^*_j)})_{j\in[n]})$ the adversary found a collision in the hash function. If H is modeled with the random oracle then the collision resistance is with probability $t(t+1)/2^{\chi+1}$. □

6 Constructing linear PCPs and MIPs

In this section, we show how to adapt known linear PCPs into the information-theoretic objects underlying our dv-SNARG constructions. In Section 6.1 we show how to transform 2-query linear PCPs into 3-prover strong linear MIPs, and in Section 6.2, we show that linear PCPs can be transformed into modded bounded LPCPs.

6.1 Linear PCPs to strong linear MIPs

[IKO07] show that linear PCPs can be transformed into strong linear MIPs. However, their transformation garners a large loss in soundness error, and so requires many repetitions to achieve specific soundness errors. We give an alternate transformation for LPCPs with 2 queries which is more efficient in practice:

Lemma 6.1. *Let $\mathcal{R}$ be a relation with a smooth LPCP over $\mathbb{F}_p$ with length ℓ, query complexity 2, and knowledge soundness κ against affine strategies. There exists a strong LMIP for $\mathcal{R}$ over $\mathbb{F}_p$ with length ℓ, query complexity 3, and strong knowledge soundness $\frac{7}{9} + \frac{36}{505-729\cdot\kappa}$. If the LPCP is input-oblivious, then so is the strong LMIP. If the LPCP is B-bounded with error α, then the strong LMIP is $(B + 4p\sqrt{\ell\lambda})$-bounded with error $\alpha + 2^{-\lambda}$.*

Proof. We begin by showing that an LPCP can be made smooth with the addition of a single query:

Claim 6.2. Let $\mathcal{R}$ be a relation with an LPCP over $\mathbb{F}_p$ with length ℓ, query complexity q, and knowledge error κ against affine strategies. Then $\mathcal{R}$ has a smooth LPCP over $\mathbb{F}_p$ with length ℓ, query complexity $q+1$, and knowledge error κ against affine strategies. If the LPCP is input-oblivious, then so is the smooth LPCP. If the LPCP is B-bounded with error α, then for every $\lambda \in \mathbb{N}$, the smooth LPCP is $(B + p^2\sqrt{\ell\lambda})$-bounded with error $\alpha + 2^{-\lambda}$.

Proof. Let $(\mathbf{P}, (V_Q, V_D))$ be the linear PCP for $\mathcal{R}$. We give a smooth linear PCP $(\mathbf{P}', (V'_Q, V'_D))$ for $\mathcal{R}$:

- $\mathbf{P}'(x, w)$: Output $\pi \leftarrow \mathbf{P}(x, w)$.
- $V'_Q(x)$:

1. Sample $(\mathsf{st}, \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_Q(x)$ and $\mathbf{a}'_0 \leftarrow_\$ \mathbb{F}_p^\ell$.
2. Output $(\mathsf{st}, \mathbf{a}'_0, \mathbf{a}'_1, \ldots, \mathbf{a}'_q)$ where $\mathbf{a}'_i = \mathbf{a}_i + \mathbf{a}'_0$.

- $V'_D(\mathsf{st}, x, b'_0, \ldots, b'_q)$: Output $b \leftarrow V_D(\mathsf{st}, x, b_1, \ldots, b_q)$ where $b_i = b'_i - b'_0$.

The linear PCP $(\mathbf{P}', (V'_Q, V'_D))$ is smooth: $\mathbf{a}'_0$ is uniform over $\mathbb{F}^\ell$, and so $\mathbf{a}'_i = \mathbf{a}_i + \mathbf{a}'_0$ is 1-wise uniform over $\mathbb{F}_p^\ell$. Observe that for every affine prover strategy $\pi \in \mathbb{F}_p^\ell$ and $c_0, \ldots, c_q \in \mathbb{F}_p$:

$$b_i = b'_i - b'_0 = \langle \pi, \mathbf{a}_i + \mathbf{a}'_0 \rangle + c_i - \langle \pi, \mathbf{a}'_0 \rangle - c_0 = \langle \pi, \mathbf{a}_i \rangle + c_i - c_0 \ .$$

Hence we can translate any prover strategy $\pi, c_0, \ldots, c_q$ in the new protocol to a prover strategy $\pi, (c_1 - c_0), \ldots, (c_q - c_0)$ in the original proof. Moreover, any set of queries $(\mathbf{a}'_0, \ldots, \mathbf{a}'_q)$ in the new scheme can be translated into $(\mathbf{a}_1, \ldots, \mathbf{a}_q)$, where the distribution of a random set of these queries is the identical to the original protocol. Completeness and knowledge soundness follow immediately from this fact.

To see that the smooth LMIP is $2B$-bounded, notice that

$$|\langle \mathbb{Z}(\pi), \mathbb{Z}(\mathbf{a}'_i) \rangle| = |\langle \mathbb{Z}(\pi), \mathbb{Z}(\mathbf{a}_i + \mathbf{a}'_0) \rangle| \le |\langle \mathbb{Z}(\pi), \mathbb{Z}(\mathbf{a}_i) \rangle| + |\langle \mathbb{Z}(\pi), \mathbb{Z}(\mathbf{a}'_0) \rangle| \ ,$$

Since the PCP is bounded, $|\langle \mathbb{Z}(\pi), \mathbb{Z}(\mathbf{a}_i) \rangle| > B$ with probability at most α. By Hoeffding's inequality, since $\langle \mathbb{Z}(\pi), \mathbb{Z}(\mathbf{a}'_0) \rangle \in [-\ell \cdot p^2, \ell \cdot p^2]$ for every $t > 0$ we have

$$\Pr[|\langle \mathbb{Z}(\pi), \mathbb{Z}(\mathbf{a}'_0)) \rangle| > t] < 2 \cdot \exp\left(-\frac{2 \cdot t^2}{\ell^2 p^4}\right)$$

Setting $t = p^2\sqrt{\ell\lambda}$, we get that we have that $|\langle \mathbb{Z}(\pi), \mathbb{Z}(\mathbf{a}'_0)) \rangle| > p^2\sqrt{\ell\lambda}$ with probability at most $2^{-2\lambda+1} < 2^{-\lambda}$. Thus, $|\langle \mathbb{Z}(\pi), \mathbb{Z}(\mathbf{a}'_i) \rangle| > B + p^2\sqrt{\ell\lambda}$ with probability at most $\alpha + 2^{-\lambda}$. □

Given Claim 6.2, we can safely assume that our linear PCP is smooth with 3 queries. We use this to construct a 3-query strong LMIP. Let $(\mathbf{P}, (V_Q, V_D))$ be the smooth LPCP. We design a 3-query strong LMIP $(\mathbf{P}', (V'_Q, V'_D))$:

- $\mathbf{P}'(x, w)$: Output $\pi \leftarrow \mathbf{P}(x, w)$.
- $V'_Q(x)$: Sample $b \leftarrow \{0, 1\}$ from the Bernoulli distribution where 0 is sampled with probability $\beta = \frac{162}{505 - 729 \cdot \kappa}$. Then, do the following:
  1. If $b = 0$: Sample $(\mathsf{st}, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3) \leftarrow V_D(x)$. Output $(\mathsf{st}', \mathbf{a}'_1, \mathbf{a}'_2, \mathbf{a}'_3)$ where $\mathsf{st}' = (b, \mathsf{st})$ and $\mathbf{a}'_i = \mathbf{a}_i$.
  2. If $b = 1$: Sample $\mathbf{z}_1, \mathbf{z}_2 \leftarrow_\$ \mathbb{F}^\ell$ and set $\mathbf{z}_3 = \mathbf{z}_1 + \mathbf{z}_2$. Output $(\mathsf{st}', \mathbf{a}'_1, \mathbf{a}'_2, \mathbf{a}'_3)$ where $\mathsf{st}' = (b, \perp)$ and $\mathbf{a}'_i = \mathbf{z}_i$.
- $V'_D(\mathsf{st}', x, b'_1, b'_2, b'_3)$: Parse $\mathsf{st}' = (b, \mathsf{st})$, and
  1. If $b = 0$: Output 1 if and only if $V_D(\mathsf{st}, x, b'_1, b'_2, b'_3) = 1$.
  2. If $b = 1$: Output 1 if and only if $b'_1 + b'_2 = b'_3$.

Completeness holds by perfect completeness of the smooth linear PCP in the case of $b = 0$, and by linearity of the honest prover strategy in the case of $b = 1$. We now prove knowledge soundness. Let Ext be the extractor for the LPCP. We construct an extractor Ext' as follows:

1. On input x and given oracle access to functions $f_1, f_2, f_3 \colon \mathbb{F}_p^\ell \to \mathbb{F}_p$.
2. For every $i \in [3]$ extract from f_i an affine function π_i, c_i by doing standard local correction of affine functions for distance at most $2/9$ (note this can be done in time $\mathsf{poly}(\ell)$ with constant probability).
3. If $\pi_i \neq \pi_j$ for some i, j, then repeat the above.
4. Otherwise, output $w \leftarrow \mathsf{Ext}(x, \pi_1, c_1, c_2, c_3)$.

Fix x and functions $f_1, \ldots, f_3$ so that $\mathbf{V}'$ accepts x when given access to f_1, f_2, f_3 with probability $\kappa' > \frac{7}{9} + \frac{36}{505-729\cdot\kappa}$. It is clear that if f_1, f_2, f_3 are 2/9-close to affine shifts (c_1, c_2, c_3) of the same linear function π, where $(\pi, (c_1, c_2, c_3))$ causes $\mathbf{V}$ to accept on x with probability greater than κ, then Ext' runs in expected polynomial time and outputs w so that $(x, w) \in \mathcal{R}$. We therefore show that this holds.

Observe that:

$$\begin{aligned}
&\Pr\left[V'_D\left(\mathsf{st}', x, b'_1, b'_2, b'_3\right) = 1 \,\middle|\, \begin{array}{r} (\mathsf{st}', \mathbf{a}'_1, \mathbf{a}'_2, \mathbf{a}'_3) \leftarrow V'_Q(x) \\ b'_i = f_i(\mathbf{a}'_i) \end{array}\right] \\
&= \beta \cdot \Pr\left[V_D\left(\mathsf{st}, x, b'_1, b'_2, b'_3\right) = 1 \,\middle|\, \begin{array}{r} (\mathsf{st}, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3) \leftarrow V_Q(x) \\ b'_i = f_i(\mathbf{a}_i) \end{array}\right] \\
&+ (1-\beta) \cdot \Pr\left[b'_1 + b'_2 = b'_3 \,\middle|\, \begin{array}{r} \mathbf{z}_1, \mathbf{z}_2 \leftarrow_\$ \mathbb{F}_p^\ell \\ \mathbf{z}_3 = \mathbf{z}_1 + \mathbf{z}_3 \\ b'_i = f_i(\mathbf{z}_i) \end{array}\right] .
\end{aligned}$$

We first show that due to the linearity check, there must be linear-consistent functions that are 2/9-close to f_1, f_2, f_3. Note that linear-consistent functions is simply a specific case of a linear function with affine shifts.

Claim 6.3. There exist linear-consistent functions $g_1, g_2, g_3 \colon \mathbb{F}^\ell \to \mathbb{F}_p$ so that for every $i \in [3]$, $\delta_i = \Delta(f_i, g_i) < 2/9$.

Proof. Suppose that there are no linear-consistent functions $g_1, g_2, g_3 \colon \mathbb{F}_p^\ell \to \mathbb{F}_p$ so that for every $i \in [3]$, $\delta_i = \Delta(f_i, g_i) < 2/9$. In this case, by Theorem 3.13,

$$\Pr\left[b'_1 + b'_2 = b'_3 \,\middle|\, \begin{array}{r} \mathbf{z}_1, \mathbf{z}_2 \leftarrow_\$ \mathbb{F}^\ell \\ \mathbf{z}_3 = \mathbf{z}_1 + \mathbf{z}_3 \\ b'_i = f_i(\mathbf{z}_i) \end{array}\right] \leq 7/9 \ .$$

Therefore,

$$\Pr\left[V'_D\left(\mathsf{st}', x, b'_1, b'_2, b'_3\right) = 1 \,\middle|\, \begin{array}{r} (\mathsf{st}', \mathbf{a}'_1, \mathbf{a}'_2, \mathbf{a}'_3) \leftarrow V'_Q(x) \\ b'_i = f_i(\mathbf{a}'_i) \end{array}\right] \leq \beta + \frac{7}{9} \cdot (1-\beta) < \kappa' \ ,$$

which contradicts our assumption that $\mathbf{V}'$ accepts x when given access to f_1, f_2, f_3 with probability κ'. $\square$

Let $\pi \in \mathbb{F}_p^\ell$, and $c_1, c_2, c_3 \in \mathbb{F}$ be so that $g_i(\mathbf{a}) = \langle \pi, \mathbf{a} \rangle + c_i$ and $c_1 + c_2 = c_3$. We now show that these linear-consistent functions describe shifts of a linear function that cause $\mathbf{V}$ to accept with probability greater than κ:

Claim 6.4. $\Pr\left[V_D\left(\mathsf{st}, x, b_1, \ldots, b_q\right) = 1 \,\middle|\, \begin{array}{c} (\mathsf{st}, \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_Q(x) \\ \forall i \in [q],\ b_i = \langle \pi, \mathbf{a}_i \rangle + c_i \end{array}\right] > \kappa$.

Proof. Suppose towards contradiction that the claim did not hold. Since each query $\mathbf{a}_i$ is smooth, $b_i =$

$f_i(\mathbf{a}_i) = \langle \pi, \mathbf{a}_i \rangle + c_i$ with probability at least $1 - \delta_i \geq 7/9$. Thus,

$$\begin{aligned}
&\Pr\left[V_D\left(\mathsf{st}, x, b'_1, b'_2, b'_3\right) = 1 \;\middle|\; \begin{array}{r} (\mathsf{st}, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3) \leftarrow V_Q(x) \\ b_i = f_i(\mathbf{a}_i) \end{array} \right] \\
\leq& 1 - \prod_{i=1}^{3}(1-\delta_i) \\
&+ \Pr\left[V_D\left(\mathsf{st}, x, b'_1, b'_2, b'_3\right) = 1 \;\middle|\; \begin{array}{r} (\mathsf{st}, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3) \leftarrow V_Q(x) \\ b_i = \langle \pi, \mathbf{a}_i \rangle + c_i \end{array} \right] \cdot \prod_{i=1}^{3}(1-\delta_i) \\
\leq& 1 - \prod_{i=1}^{3}(1-\delta_i) + \kappa \cdot \prod_{i=1}^{3}(1-\delta_i) \\
\leq& 1 - \left(\frac{7}{9}\right)^3 + \kappa \ .
\end{aligned}$$

Thus, the probability that the verifier accepts is at most

$$\begin{aligned}
&\Pr\left[V'_D\left(\mathsf{st}', x, b'_1, b'_2, b'_3\right) = 1 \;\middle|\; \begin{array}{r} (\mathsf{st}', \mathbf{a}'_1, \mathbf{a}'_2, \mathbf{a}'_3) \leftarrow V'_Q(x) \\ b'_i = \mathbf{P}'_i(\mathbf{a}'_i) \end{array} \right] \\
&\leq \beta \cdot \left(1 - \left(\frac{7}{9}\right)^3 + \kappa\right) + 1 - \beta < \kappa' \ .
\end{aligned}$$

□

Claim 6.5. The strong LMIP is $(B + 4p^2\sqrt{\ell\lambda})$-bounded with error $\alpha + 2^{-\lambda}$.

Proof. We consider each choice of $b \in \{0, 1\}$.

- If $b = 0$, then the smooth linear LPCP is $(B + p^2\sqrt{\ell\lambda})$-bounded with error $\alpha + 2^{-\lambda}$.
- If $b = 1$, then two of the queries are uniformly random over $\mathbb{F}$ and the last is the sum of the two. That is, for the first and second queries, $\langle \mathbb{Z}(\pi), \mathbb{Z}(\mathbf{a}_i) \rangle$ is uniform over $[-\ell \cdot p^2, \ell \cdot p^2]$. By the Hoeffding inequality, for every $t > 0$ and $i \in \{1, 2\}$,

  $$\Pr\left[|\langle \mathbb{Z}(\pi), \mathbb{Z}(\mathbf{a}_i) \rangle| \geq t\right] \leq 2\exp\left(-\frac{t^2}{2\ell \cdot p^2}\right)$$

  For the final query, we have

  $$|\langle \mathbb{Z}(\pi), \mathbb{Z}(\mathbf{a}_3) \rangle| = |\langle \mathbb{Z}(\pi), \mathbb{Z}(\mathbf{a}_1 + \mathbf{a}_2) \rangle| \leq |\langle \mathbb{Z}(\pi), \mathbf{a}_1 \rangle + \langle \mathbb{Z}(\pi), \mathbf{a}_2 \rangle| \ ,$$

  and so when each of the queries above is within t, the final query is within bound $2t$. Taking a union bound, with probability $1 - 4\exp\left(-\frac{t^2}{2\ell \cdot p^2}\right)$, all of the queries are within bound $2t$.

  By choosing $t = 4p \cdot \sqrt{\ell\lambda}$ we get that all of the queries are within $[-4p\sqrt{\ell\lambda}, 4p\sqrt{\ell\lambda}]$ with probability at least $1 - 2^{-\lambda}$.

Putting both of these together, we have that all of the queries are within bound $B + 4p^2\sqrt{\ell\lambda}$ with probability at least $1 - \alpha - 2^{-\lambda}$. □

□

6.2 Modded LPCPs

We define modded linear PCPs, and show that any standard linear PCP can be transformed into a modded one. Modded LPCPs are linear PCPs where each query needs to be within a certain bound (over a large field) and, if it is within this bound, then the verifier receives the query answer after being modded.

Definition 6.6. A (bounded) modded linear PCP (mod-LPCP) $(\mathbf{P}, (V_Q, V_D))$ for a relation $\mathcal{R} = \{(x, w)\}$ S is defined as follows:

- *Syntax.* We describe a mod-LPCP with input length n, proof length ℓ, a big field size p', query complexity q, small field sizes $p_1, \ldots, p_q \in \mathbb{N}$, and a bound $B', B \in \mathbb{N}$ with $B' > B$:
  - The verifier query algorithm V_Q receives as input $x \in \mathbb{F}^n$. It outputs a state $\mathsf{st} \in \{0,1\}^*$, and q queries $\mathbf{a}_1, \ldots, \mathbf{a}_q \in \mathbb{F}_{p'}^{\ell}$.
  - The (honest) prover algorithm $\mathbf{P}$ receives an input $x \in \mathbb{F}^n$ and witness $w \in \mathbb{F}^h$. It outputs a proof $\pi \in \mathbb{F}_{p'}^{\ell}$.
  - The verifier decision algorithm V_D receives as input a state $\mathsf{st} \in \{0,1\}^*$, an input $x \in \mathbb{F}^n$, and query answers $b_1 \in \mathbb{F}_{p_1}, \ldots, b_q \in \mathbb{F}_{p_q}$. It outputs a bit $b \in \{0,1\}$.
- *Perfect completeness.* A mod-LPCP has perfect completeness if for all $(x, w) \in \mathcal{R}$:

$$\Pr\left[\begin{array}{c|r} \forall i \in [q], d_i \in [-B, B] & \pi \leftarrow \mathbf{P}(x, w) \\ V_D(\mathsf{st}, x, b_1, \ldots, b_q) = 1 & (\mathsf{st}, \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_Q(x) \\ & \forall i \in [q],\ d_i = \langle \pi, \mathbf{a}_i \rangle \\ & \forall i \in [q],\ b_i = \mathbb{Z}_{p_i}(d_i) \end{array}\right] = 1 \ .$$

- *Soundness.* A mod-LPCP has soundness error δ if for every $x \notin \mathcal{L}$, $\pi \in \mathbb{F}_{p'}^{\ell}$, and $c_1 \in \mathbb{F}_{p_1}, \ldots, c_q \in \mathbb{F}_{p_q}$:

$$\Pr\left[\begin{array}{c|r} \forall i \in [q], d_i \in [-B', B'] & (\mathsf{st}, \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_Q(x) \\ V_D(\mathsf{st}, x, b_1, \ldots, b_q) = 1 & \forall i \in [q],\ d_i = \langle \pi, \mathbf{a}_i \rangle \\ & \forall i \in [q],\ b_i = \mathbb{Z}_{p_i}(d_i) + c_i \end{array}\right] \leq \delta \ .$$

- *Knowledge.* A mod-LPCP satisfies knowledge soundness κ if there exists a PPT extractor Ext such that for every instance x, proof $\pi \in \mathbb{F}_{p'}^{\ell}$, and shifts $c_1 \in \mathbb{F}_{p_1}, \ldots, c_q \in \mathbb{F}_{p_q}$ if,

$$\Pr\left[\begin{array}{c|r} \forall i \in [q], d_i \in [-B', B'] & (\mathsf{st}, \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_Q(x) \\ V_D(\mathsf{st}, x, b_1, \ldots, b_q) = 1 & \forall i \in [q],\ d_i = \langle \pi, \mathbf{a}_i \rangle \\ & \forall i \in [q],\ b_i = \mathbb{Z}_{p_i}(d_i) + c_i \end{array}\right] > \kappa \ ,$$

  then $(x, \mathsf{Ext}(x, \pi, c_1, \ldots, c_q)) \in \mathcal{R}$.

We say that a mod-LPCP is **instance-independent** if $V_Q(x)$ is a function only of $|x|$, in which case we specify its input by $1^{|x|}$ (i.e., the verifier query algorithm is $V_Q(1^{|x|})$).

We show that linear PCPs can be transformed into mod-LPCP:

Lemma 6.7. *Let $\mathcal{R}$ be a relation with a LPCP over $\mathbb{F}_p$ with soundness error δ query complexity q and proof length ℓ, and let $p' > p$ be a parameter so that $\frac{p'}{2 \cdot p \cdot \ell} > 2B'$. Then $\mathcal{R}$ has a mod-LPCP over moduli p' and* $\mathbf{p}$ *with soundness error δ, $q + \lceil -\log \delta \rceil$ queries, and proof length ℓ. The mod-LPCP moduli are $p_i = p$ for $i \in [q]$ and $p_i = 1$ for $i \in [q+1, q + \lceil -\log \delta \rceil]$.*

If the LPCP is instance-oblivious, then so is the mod-LPCP. If the LPCP is B-bounded with error α, then the mod-LPCP is $\max\{B, 2\sqrt{p\ell\lambda}\}$-bounded with error $\alpha + 2^{-\lambda}\lceil \log 1/\delta \rceil$.

Remark 6.8. Notice that in the resultant mod-LPCP of Lemma 6.7, the last $\lceil -\log \delta \rceil$ queries have a modulus of 1. Therefore, V_D always receives 0 at those positions. They help only with soundness because of the bounds check but do not communicate any information to the verifier.

Proof. Let $q' = q + \lceil -\log \delta \rceil$ and let $(\mathbf{P}, (V_Q, V_D))$ be the original LPCP. We define a mod-LPCP $(\mathbf{P}', (V'_Q, V'_D))$ im the following construction:

- $\mathbf{P}'(x, w)$:
  1. Let $\pi \leftarrow \mathbf{P}(x, w) \in \mathbb{F}_p^\ell$.
  2. Output $\pi' = \mathbb{Z}_{p'}(\pi) \in \mathbb{F}_{p'}^\ell$.
- $V'_Q(x)$:
  1. Let $(\mathsf{st}, \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_Q(x)$.
  2. We define a distribution $\mathcal{D}$ whose image is $\{-1, 0, 1\}$ which outputs:
     - -1 with probability $1/4$;
     - 1 with probability $1/4$;
     - 0 with probability $1/2$.
  3. For $i \in [\lceil -\log \delta \rceil]$ sample $\mathbf{a}_{q+i} \leftarrow \mathcal{D}^\ell$.
  4. Output queries $(\mathbf{a}'_i)_{i \in [q']} = (\mathbb{Z}_{p'}(\mathbf{a}_i))_{i \in [q']}$.
- $V'_D(\mathsf{st}, x, b_1, \ldots, b_{q'})$:
  1. Output $b \leftarrow V_D(\mathsf{st}, x, b_1, \ldots, b_q)$. (Note that the verifier ignores b_i for $i > q$.)

Completeness follows by the construction, we prove soundness and later explain knowledge soundness. To prove soundness we distinguish two cases:

- If for every $j \in [\ell]$, we have $|\pi[j]| < \frac{p'}{2 \cdot p \cdot \ell}$. In this case, for all $i \in [q]$, the magnitude of $\langle \pi, \mathbf{a}_i \rangle$ when computations are done over the integers (rather than p') is small:

$$|\langle \mathbb{Z}(\pi), \mathbb{Z}(\mathbf{a}'_i) \rangle| = \left| \sum_{j \in [\ell]} \mathbb{Z}(\pi[j]) \cdot \mathbb{Z}(\mathbf{a}_i[j]) \right| \leq \sum_{j \in [\ell]} |\mathbb{Z}(\pi[j])| \cdot |\mathbb{Z}(\mathbf{a}_i[j])|$$
$$< \sum_{j \in [\ell]} \frac{p'}{2 \cdot p \cdot \ell} \cdot p = p'/2 \ .$$

  The above holds by triangle inequality. Since over the integers, the inner product is smaller than p', there is no wrap-around when computing over the integers versus over $\mathbb{Z}_{p'}$. Thus,

$$\mathbb{Z}_p(\langle \pi, \mathbf{a}'_i \rangle) = \mathbb{Z}_p(\langle \mathbb{Z}(\pi), \mathbb{Z}(\mathbf{a}'_i) \rangle) = \mathbb{Z}_p\left(\sum_{j \in [\ell]} \mathbb{Z}(\pi[j]) \cdot \mathbb{Z}(\mathbf{a}'_i[j]) \right)$$
$$= \sum_{j \in [\ell]} \mathbb{Z}_p(\pi[j]) \cdot \mathbb{Z}_p(\mathbf{a}'_i[j]) = \langle \mathbb{Z}_p(\pi), \mathbb{Z}_p(\mathbf{a}'_i) \rangle = \langle \mathbb{Z}_p(\pi), \mathbf{a}_i \rangle$$

  Therefore,

$$V'_D(\mathsf{st}, x, \mathbb{Z}_p(\langle \pi, \mathbf{a}'_1 \rangle) + c_1, \ldots, \mathbb{Z}_p(\langle \pi, \mathbf{a}'_{q'} \rangle) + c_{q'})$$
$$= V_D(\mathsf{st}, x, \langle \mathbb{Z}_p(\pi), \mathbf{a}_1 \rangle + c_1, \ldots, \langle \mathbb{Z}_p(\pi), \mathbf{a}_q \rangle + c_q) \ .$$

  Thus, this case reduces to soundness of $(\mathbf{P}, (V_Q, V_D))$ against the proof $(\mathbb{Z}_p(\pi), c_1, \ldots, c_q)$.

- If there exists $j \in [\ell]$ such that $|\pi[j]| \geq \frac{p'}{2 \cdot p \cdot \ell}$. Fix this j for the remainder of the proof. For the rest of the analysis we relax the malicious prover's winning condition: the prover wins if for every $i \in [q+1, q']$ it holds that $\langle \pi, \mathbf{a}_i \rangle \in [-B', B']$. Since we have removed restrictions from the probability that we need to bound, this can only increase the probability. Thus by bounding this probability, we bound the probability that the results are within range and the verifier accepts.

Fix an arbitrary $i \in [q+1, q']$. We analyze the probability that $\langle \pi, \mathbf{a}_i \rangle \in [-B', B']$. By opening up the definition, $\langle \pi, \mathbf{a}_i \rangle = \pi[j] \cdot \mathbf{a}_i[j] + \sum_{j' \in [\ell] \setminus \{j\}} \pi[j'] \cdot \mathbf{a}_i[j']$. We show that for every $s \in \mathbb{F}_{p'}$ we show that

$$\Pr_{\mathbf{a}_i[j] \leftarrow \mathcal{D}}[(\pi[j] \cdot \mathbf{a}_i[j] + s_i) \in [-B', B']] \leq 1/2 \ .$$

Fix $s \in \mathbb{F}_{p'}$ and recall that $|\pi[j]| > \frac{p'}{2 \cdot p \cdot \ell} > 2B'$ and that $\mathbf{a}_i[j]$ is chosen from $\mathcal{D}$. Then,

- If $|s| \leq B'$ then $|\pi[j] \cdot 1 + s| > B'$ and $|\pi[j] \cdot -1 + s| > B'$. Since $\mathbf{a}_i[j] \in \{-1, 1\}$ with probability 1/2, we have that $|\pi[j] \cdot \mathbf{a}_i[j] + s| > B'$ with probability 1/2.
- If $|s| > B'$, then $|\pi[j] \cdot 0 + s| > B'$. Since $\mathbf{a}_i[j] = 0$ with probability 1/2, we have that $|\pi[j] \cdot \mathbf{a}_i[j] + s| > B'$ with probability 1/2.

Thus, we have that

$$\Pr_{\mathbf{a}_i \leftarrow \mathcal{D}^\ell}[\langle \pi, \mathbf{a}_i \rangle \in [-B', B']] = \Pr\left[(\pi[j] \cdot \mathbf{a}_i[j] + s) \in [-B', B'] \;\middle|\; \begin{array}{r} \forall j' \in [\ell] \setminus \{j\} \mathbf{a}_i[j'] \leftarrow \mathcal{D} \\ s = \sum_{j' \in [\ell] \setminus \{j\}} \pi[j'] \cdot \mathbf{a}_i[j'] \\ \mathbf{a}_i[j] \leftarrow \mathcal{D} \end{array} \right] \leq 1/2$$

Finally, since the queries $\mathbf{a}_{q+1}, \ldots, \mathbf{a}_{q'}$ are all chosen independently,

$$\begin{aligned} \Pr[\forall i \in [q'], \langle \pi, \mathbf{a}_i \rangle \in [-B', B']] &\leq \Pr[\forall i \in [q+1, q'], \langle \pi, \mathbf{a}_i \rangle \in [-B', B']] \\ &\leq 2^{-\lceil -\log \delta \rceil} \\ &\leq \delta. \end{aligned}$$

Remark 6.9. If $(\mathbf{P}, (V_Q, V_D))$ has knowledge soundness β then $(\mathbf{P}', (V'_Q, V'_D))$ also has knowledge soundness β. This follows by the exact same argument as above using the same extractor as the original LPCP.

Claim 6.10. The mod-PCP is $\max\{B, 2\sqrt{p\ell\lambda}\}$-bounded with error $\alpha + 2^{-\lambda}\lceil \log 1/\delta \rceil$.

Proof. The first q queries are B-bounded with error α by assumption. Each for $\mathbf{a}_i$, each entry $\mathbf{a}_i[j]$ is bounded within $[-1, 1]$ and has expectation 0. Thus, $|\langle \mathbb{Z}(\pi), \mathbb{Z}(\mathbf{a}_i) \rangle| \leq p \cdot \ell$ and has an expectation of 0. By Hoeffding's inequality,

$$\Pr[|\langle \mathbb{Z}(\pi), \mathbb{Z}(\mathbf{a}_i) \rangle| > t] < 2 \cdot \exp\left(-\frac{t^2}{2p^2\ell^2}\right) \ .$$

Since there are $\lceil \log 1/\delta \rceil$ such queries, the probability that there exists a value that is out-of-bound t is at most $\lceil \log 1/\delta \rceil \cdot 2 \cdot \exp\left(-\frac{t^2}{2p^2\ell^2}\right)$. Setting $t = 2\sqrt{p\ell\lambda}$, we get that all of these are within bounds with probability at least $1 - 2^{-\lambda}\lceil \log 1/\delta \rceil$.

Thus, all together, we get the bound $\max\{B, 2\sqrt{p\ell\lambda}\}$ with error $\alpha + 2^{-\lambda}\lceil \log 1/\delta \rceil$. □

□

7 Designated-verifier SNARGs from compressible encryption

In this section, we show how to construct SNARGs (in fact, SNARKs) by combining compressible encryption schemes with suitable information-theoretic protocols. In Section 7.1, we show this from isolated homomorphism (Definition 5.1) and strong MIPs, and in Section 7.2, we show this from bound-limited homomorphism (Definition 5.2) and modded LPCPs.

We state here the dv-SNARKs resultant as corollaries from our proofs and transformations.

- Derived by combining Lemma 7.3 with the packed ElGamal encryption scheme and a strong linear MIP and making enough repetitions:

  Corollary 7.1 (Dv-SNARKs from packed ElGamal). *Let $C \colon \mathbb{F}_p^n \times \mathbb{F}_p^h \to \mathbb{F}_p$ be an arithmetic circuit of size s. For every $\lambda, \tau \in \mathbb{N}$ there are dv-SNARKs for $\mathcal{R}_C = \{(x,w) \in \mathbb{F}^n \times \mathbb{F}^h \mid C(x,w) = 1\}$ in the GGM with of group size 2^λ with completeness error $\mathsf{negl}(\lambda)$, and the following parameters:*

  (Linear CRS.) *Using the LMIP of Corollary 3.10:*

  - *Message length: $1\mathbb{G}$ element and $O(\tau)$ bits;*
  - *Knowledge soundness: $2^{-\tau} + 8t^2/2^\lambda$ against t-query adversaries;*
  - *CRS length: $O(\tau \cdot (s + \mathsf{poly}(p)))$ $\mathbb{G}$ elements;*
  - *Setup time: $O(\tau \cdot (s + \mathsf{poly}(p)))$;*
  - *Prover expected runtime: $\tilde{O}(\lambda s \tau^2 \mathsf{poly}(p))$;*
  - *Verifier runtime: $\tilde{O}(\lambda s \tau^2 p^2)$.*

  (Concrete efficiency.) *Using the LMIP derived by combining Theorem 3.8 and Lemma 6.1):*

  - *Message length: $1\mathbb{G}$ element and $\lceil 3\tau \log_2 p \rceil$ bits;*
  - *Knowledge soundness: $(\frac{7}{9} + \frac{36p}{505p-1458})^\tau + 8t^2/2^\lambda$ against t-query adversaries;*
  - *CRS length: $(3\tau + 1) \cdot (s^2 + s)$ $\mathbb{G}$ elements;*
  - *Setup time: $O(\tau s^2)$;*
  - *Prover expected runtime: $\tilde{O}(\lambda s^2 \tau^2 p^2)$;*
  - *Verifier runtime: $\tilde{O}(\lambda s^2 \tau^2 p^2)$ ($\tilde{O}(\lambda s \tau^2 p^2)$ if we restrict to Boolean circuits).*

  Above, the knowledge soundness errors hold for $t > 4 \cdot t_{\mathbf{V}} + 1$ where $t_{\mathbf{V}}$ is the verification time.

- Derived by combining Lemma 7.5 with the packed ElGamal with hash scheme and a linear PCP transformed into a modded LPCP using Lemma 6.7 we get dv-SNARKs where the number of added bits approaches 2τ:

  Corollary 7.2 (Dv-SNARKs from packed ElGamal with hash). *Let $C \colon \mathbb{F}_p^n \times \mathbb{F}_p^h \to \mathbb{F}_p$ be an arithmetic circuit of size s with $p > 2$. For every $\lambda, \tau \in \mathbb{N}$ there are dv-SNARKs for the relation $\mathcal{R}_C = \{(x,w) \in \mathbb{F}^n \times \mathbb{F}^h \mid C(x,w) = 1\}$ in the GGM with group size 2^λ and with random oracle output length λ with completeness error $\mathsf{negl}(\lambda)$ and the following parameters:*

  (Linear CRS.) *Using the LPCP of Theorem 3.9:*

  - *Message length: $1\mathbb{G}$ element, 1 ROM output (λ bits), and $\lceil \tau \log p \rceil$ bits;*
  - *Knowledge soundness: $O(p^{-\tau/2}) + 10t^2/2^\lambda$ against t-query adversaries (to both the ROM and GGM);*
  - *CRS length: $O(\tau \cdot (s + \mathsf{poly}(p)))$ $\mathbb{G}$ elements;*
  - *Setup time: $O(\tau \cdot (s + \mathsf{poly}(p)))$;*
  - *Prover expected runtime: $\tilde{O}(\lambda s \tau^2 \mathsf{poly}(p))$;*
  - *Verifier runtime: $\tilde{O}(\lambda s \tau^2 p^2)$.*

  (Minimal length.) *Using the LPCP of Theorem 3.8:*

  - *Message length: $1\mathbb{G}$ element, 1 ROM output (λ bits), and $\lceil 2\tau \log p \rceil$ bits;*
  - *Knowledge soundness: $(2/p)^{-\tau} + 10t^2/2^\lambda$ against t-query adversaries (to both the ROM and GGM);*
  - *CRS length: $O(\tau s^2)$ $\mathbb{G}$ elements;*
  - *Setup time: $O(\tau s^2)$;*
  - *Prover expected runtime: $\tilde{O}(\lambda s^2 \tau^2 p^2)$;*
  - *Verifier runtime: $\tilde{O}(\lambda s^2 \tau^2 p^2)$ ($\tilde{O}(\lambda s \tau^2 p^2)$ if we restrict to Boolean circuits).*

  Above, the knowledge soundness errors hold for $t > 4 \cdot t_{\mathbf{V}} + 1$ where $t_{\mathbf{V}}$ is the verification time.

7.1 Construction from isolated homomorphism

We show how to combine a strong LMIP with knowledge soundness, and an isolated homomorphic encryption scheme into a dv-SNARK:

Lemma 7.3. *Suppose the existence of the following ingredients:*

- *An input-oblivious strong linear MIP over finite field $\mathbb{F}_p$ for a relation $\mathcal{R}$ that is B-bounded with error α, soundness δ, knowledge soundness κ, q queries, query length ℓ, prover running time t_P, and verifier running time (t_Q, t_D).*
- *A compressible linearly homomorphic encryption scheme with q slots, plaintext moduli p, ciphertext size σ_{ct}, compressed ciphertext size σ_{cct}, decryption bound B, and encryption, compression, evaluation, and decryption times $(t_{\mathsf{enc}}, t_{\mathsf{cmp}}, t_{\mathsf{eval}}, t_{\mathsf{dec}})$. Furthermore, let $\varepsilon_{\mathsf{cor}}$ be its correctness error, $\varepsilon_{\mathsf{ih}}$ be its isolated homomorphism distinguishability error and $\varepsilon_{\mathsf{sem}}$ be its semantic security advantage.*

Then there is a designated-verifier SNARK for $\mathcal{R}$ with:

- *Completeness error: $\alpha + \ell \cdot \varepsilon_{\mathsf{cor}}(\lambda)$;*
- *Soundness: $\delta + \varepsilon_{\mathsf{ih}}(\lambda, t, \ell) + \varepsilon_{\mathsf{sem}}(\lambda, t, \ell)$ against t-query adversaries;*
- *Knowledge soundness: $\kappa + \varepsilon_{\mathsf{ih}}(\lambda, t, \ell) + \ell \cdot \varepsilon_{\mathsf{sem}}(\lambda, t)$ against t-query adversaries;*
- *Message length: $\sigma_{\mathsf{cct}}(\lambda)$;*
- *CRS length: $\ell \cdot \sigma_{\mathsf{ct}}(\lambda)$;*
- *Setup time: $O(t_Q + \ell \cdot t_{\mathsf{enc}}(\lambda))$;*
- *Prover runtime: $O(t_P + t_{\mathsf{eval}}(\lambda, \ell) + t_{\mathsf{cmp}}(\lambda))$;*
- *Verifier runtime: $O(t_D + t_{\mathsf{dec}}(\lambda))$.*

Proof. Let $(\mathbf{P}, (V_Q, V_D))$ be the linear MIP, and $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ be the homomorphic encryption scheme. We construct a SNARK

Construction 7.4. The SNARK $(\mathsf{Setup}, \mathbf{P}', \mathbf{V}')$ is defined as follows:

- $\mathsf{Setup}^{\mathcal{G}}(1^n)$:
  1. Sample $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}}$ and $(\mathsf{st}', \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_Q(1^n)$.
  2. For every $i \in [\ell]$ let $\mathbf{z}_i = (\mathbf{a}_1[i], \ldots, \mathbf{a}_q[i]) \in \mathbb{F}^q$ and compute $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathbf{z}_i)$.
  3. Output $\mathsf{crs} := (\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$ and $\mathsf{st} := (\mathsf{sk}, \mathsf{st}')$.
- $\mathbf{P}'^{\mathcal{G}}(\mathsf{crs}, x, w)$:
  1. Parse $\mathsf{crs} := (\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$ and compute $\pi \leftarrow \mathbf{P}(x, w)$.
  2. Compute $\mathsf{ct}' \leftarrow \mathsf{Eval}^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell, \pi)$.
  3. Let $\mathsf{cct} := \mathsf{Compress}^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}')$.
  4. Output $\mathsf{pf} := \mathsf{cct}$.
- $\mathbf{V}'^{\mathcal{G}}(\mathsf{st}, x, \mathsf{pf})$:
  1. Parse $\mathsf{st} = (\mathsf{sk}, \mathsf{st}')$ and $\mathsf{pf} = \mathsf{cct}$.
  2. If $\mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct}) = \perp$, then output 0. Otherwise, let $(b_1, \ldots, b_q) = \mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct})$.
  3. Output 1 if $V_D(\mathsf{st}', x, b_1, \ldots, b_q) = 1$ and otherwise output 0.

Completeness, and the complexity parameters follow immediately from the construction. We prove soundness and knowledge by first proving soundness, and then explaining the (small) differences when proving knowledge.

Soundness. Fix λ and a prover $\mathbf{P}'$ for the dv-SNARK soundness experiment that makes t queries. We show that the probability that $\mathbf{V}'$ outputs 1 is at most $\delta + \varepsilon_{\mathsf{sem}} + \varepsilon_{\mathsf{ih}}$. Suppose towards contradiction to the soundness error of the strong LMIP that $\mathbf{P}'$ causes the verifier to accept with probability $\delta' > \delta + \varepsilon_{\mathsf{sem}} + \varepsilon_{\mathsf{ih}}$.

Let $\mathcal{S}$ be the simulator of the isolated homomorphism experiment of the encryption scheme, and define a plaintext generator $\mathcal{T}$ and adversary $\mathcal{A}$:

- $\mathcal{T}^{\mathcal{G}}(\mathsf{pk})$: Sample $(\mathsf{st}', \mathbf{a}_1, \dots, \mathbf{a}_1) \leftarrow V_Q(x)$. Output $(\mathsf{st}', (\mathbf{z}_1, \dots, \mathbf{z}_\ell))$ where $\mathbf{z}_i = (\mathbf{a}_1[i], \dots, \mathbf{a}_q[i])$.
- $\mathcal{A}^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \dots, \mathsf{ct}_\ell)$: Let $\mathsf{crs} = (\mathsf{pk}, \mathsf{ct}_1, \dots, \mathsf{ct}_\ell)$ and compute and output $(x, \mathsf{cct}) \leftarrow \mathbf{P}'^{\mathcal{G}}(\mathsf{crs})$.

By plugging in to the isolated homomorphism experiment this plaintext generator and adversary, the outputs of the following two experiments have statistical distance at most $\varepsilon_{\mathsf{ih}}$:

- **Real**:
  1. Sample $\mathcal{G} \leftarrow \mathsf{GGM}(\lambda)$.
  2. Let $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}}$.
  3. $(\mathsf{st}', \mathbf{a}_1, \dots, \mathbf{a}_\ell) \leftarrow V_Q(1^n)$.
  4. $\mathbf{z}_i = (\mathbf{a}_1[i], \dots, \mathbf{a}_q[i])$ for all $i \in [\ell]$.
  5. $\mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, \mathbf{z}_i)$ for all $i \in [\ell]$.
  6. $(x, \mathsf{cct}) \leftarrow \mathbf{P}'^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \dots, \mathsf{ct}_\ell)$.
  7. If $\mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct}) = \perp$, output $\perp$.
  8. $(b_1, \dots, b_q) \leftarrow \mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct})$
  9. Output $(\mathsf{st}', x, b_1, \dots, b_q)$.
- **Ideal**:
  1. Sample $\mathcal{G} \leftarrow \mathsf{GGM}(\lambda)$.
  2. Let $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}}$.
  3. $(\mathsf{st}', \mathbf{a}_1, \dots, \mathbf{a}_\ell) \leftarrow V_Q(1^n)$.
  4. $\mathbf{z}_i = (\mathbf{a}_1[i], \dots, \mathbf{a}_q[i])$ for all $i \in [\ell]$.
  5. $\mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, \mathbf{z}_i)$ for all $i \in [m]$.
  6. $(\mathsf{cct}, x) \xleftarrow{\mathsf{tr}} \mathbf{P}'^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \dots, \mathsf{ct}_\ell)$.
  7. $(f_1, \dots, f_q) \leftarrow \mathcal{S}(\mathsf{tr}, \mathsf{pk}, \mathsf{ct}_1, \dots, \mathsf{ct}_\ell, \mathsf{cct})$.
  8. If $\mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct}) = \perp$, output $\perp$.
  9. $b_i = f_i(\mathbf{a}_i)$ for all $i \in [q]$.
  10. Output $(\mathsf{st}', x, b_1, \dots, b_q)$.

Consider the following predicate $p(X)$: if $X = \perp$ or X cannot be parsed as $X = (\mathsf{st}', x, b_1, \dots, b_q)$, then output 0. Otherwise, output 1 if $|x| = n$, $x \in \mathcal{L}(\mathcal{R})$, and $V_D(\mathsf{st}', x, b_1, \dots, b_q) = 1$, and otherwise output 0. Observe that after applying the predicate to the output of the real experiment, we get the predicate of whether the SNARK verifier accepted in the adaptive soundness experiment, which happens with probability δ'. Thus, by $\varepsilon_{\mathsf{ih}}$ statistical indistinguishability of the real and ideal games, the probability of the predicate being satisfied in the ideal game is at least $\delta' - \varepsilon_{\mathsf{ih}}$:

$$\Pr\left[\begin{array}{c|r} & \mathcal{G} \leftarrow \mathsf{GGM}(\lambda) \\ & (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}} \\ x \notin \mathcal{L}(\mathcal{R}) & (\mathsf{st}', \mathbf{a}_1, \dots, \mathbf{a}_q) \leftarrow V_Q(1^n) \\ \wedge\ \mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct}) \neq \perp & \forall i \in [\ell],\ \mathbf{z}_i = (\mathbf{a}_1[i], \dots, \mathbf{a}_q[i]) \\ \wedge\ V_D(\mathsf{st}', x, b_1, \dots, b_q) = 1 & \forall i \in [\ell],\ \mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, \mathbf{z}_i) \\ & (\mathsf{cct}, x) \xleftarrow{\mathsf{tr}} \mathbf{P}'^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \dots, \mathsf{ct}_\ell) \\ & (f_1, \dots, f_q) \leftarrow \mathcal{S}(\mathsf{tr}, \mathsf{pk}, \mathsf{ct}_1, \dots, \mathsf{ct}_\ell, \mathsf{cct}) \\ & \forall i \in [q],\ b_i = f_i(\mathbf{a}_i) \end{array}\right] \geq \delta' - \varepsilon_{\mathsf{ih}}(\lambda, t, \ell)\ .$$

We can now remove the check that decryption is done correctly, thus making the predicate independent of the encryption secret key. That is, the expression above is bounded from above by

$$\Pr\left[\begin{array}{c|r} x \notin \mathcal{L}(\mathcal{R}) & \mathcal{G} \leftarrow \mathsf{GGM}(\lambda) \\ \wedge\, V_D(\mathsf{st}', x, b_1, \dots, b_q) = 1 & (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}} \\ & (\mathsf{st}', \mathbf{a}_1, \dots, \mathbf{a}_q) \leftarrow V_Q(1^n) \\ & \forall i \in [\ell],\ \mathbf{z}_i = (\mathbf{a}_1[i], \dots, \mathbf{a}_q[i]) \\ & \forall i \in [\ell],\ \mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, \mathbf{z}_i) \\ & (\mathsf{cct}, x) \xleftarrow{\mathsf{tr}} \mathbf{P}'^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \dots, \mathsf{ct}_\ell) \\ & (f_1, \dots, f_q) \leftarrow \mathcal{S}(\mathsf{tr}, \mathsf{pk}, \mathsf{ct}_1, \dots, \mathsf{ct}_\ell, \mathsf{cct}) \\ & \forall i \in [q],\ b_i = f_i(\mathbf{a}_i) \end{array}\right]$$

We now utilize semantic security of the encryption scheme to change the encryptions to 0^q.

$$\Pr\left[\begin{array}{c|r} x \notin \mathcal{L}(\mathcal{R}) & \mathcal{G} \leftarrow \mathsf{GGM}(\lambda) \\ \wedge\, V_D(\mathsf{st}', x, b_1, \dots, b_q) = 1 & (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}} \\ & (\mathsf{st}', \mathbf{a}_1, \dots, \mathbf{a}_q) \leftarrow V_Q(1^n) \\ & \forall i \in [\ell],\ \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, 0^q) \\ & (\mathsf{cct}, x) \xleftarrow{\mathsf{tr}} \mathbf{P}'^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \dots, \mathsf{ct}_\ell) \\ & (f_1, \dots, f_q) \leftarrow \mathcal{S}(\mathsf{tr}, \mathsf{pk}, \mathsf{ct}_1, \dots, \mathsf{ct}_\ell, \mathsf{cct}) \\ & \forall i \in [q],\ b_i = f_i(\mathbf{a}_i) \end{array}\right]$$

$$\geq \Pr\left[\begin{array}{c|r} x \notin \mathcal{L}(\mathcal{R}) & \mathcal{G} \leftarrow \mathsf{GGM}(\lambda) \\ \wedge\, V_D(\mathsf{st}', x, b_1, \dots, b_q) = 1 & (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}} \\ & (\mathsf{st}', \mathbf{a}_1, \dots, \mathbf{a}_q) \leftarrow V_Q(1^n) \\ & \forall i \in [\ell],\ \mathbf{z}_i = (\mathbf{a}_1[i], \dots, \mathbf{a}_q[i]) \\ & \forall i \in [\ell],\ \mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, \mathbf{z}_i) \\ & (\mathsf{cct}, x) \xleftarrow{\mathsf{tr}} \mathbf{P}'^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \dots, \mathsf{ct}_\ell) \\ & (f_1, \dots, f_q) \leftarrow \mathcal{S}(\mathcal{A}, \mathsf{pk}, \mathsf{ct}_1, \dots, \mathsf{ct}_\ell) \\ & \forall i \in [q],\ b_i = f_i(\mathbf{a}_i) \end{array}\right] - \varepsilon_{\mathsf{sem}}(\lambda, t, \ell) \ .$$

Indeed, otherwise we could run the above experiments to distinguish ciphertexts of $(\mathbf{z}_1, \dots, \mathbf{z}_\ell)$ from ciphertexts of the all-zeroes string. Thus, we have that

$$\Pr\left[\begin{array}{c|r} x \notin \mathcal{L}(\mathcal{R}) & \mathcal{G} \leftarrow \mathsf{GGM}(\lambda) \\ \wedge\, V_D(\mathsf{st}', x, b_1, \dots, b_q) = 1 & (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}} \\ & (\mathsf{st}', \mathbf{a}_1, \dots, \mathbf{a}_q) \leftarrow V_Q(1^n) \\ & \forall i \in [\ell],\ \mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, 0^q) \\ & (\mathsf{cct}, x) \xleftarrow{\mathsf{tr}} \mathbf{P}'^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \dots, \mathsf{ct}_\ell) \\ & (f_1, \dots, f_q) \leftarrow \mathcal{S}(\mathsf{tr}, \mathsf{pk}, \mathsf{ct}_1, \dots, \mathsf{ct}_\ell, \mathsf{cct}) \\ & \forall i \in [q],\ b_i = f_i(\mathbf{a}_i) \end{array}\right] \geq \delta' - \varepsilon_{\mathsf{ih}}(\lambda, \ell, t) - \varepsilon_{\mathsf{sem}}(\lambda, t, \ell) > \delta \ .$$

Observe that, now the choice of x, $f_1, \dots, f_q$ does not depend on $\mathbf{a}_1, \dots, \mathbf{a}_q$. By an averaging argument, there exist x, $f_1, \dots, f_q$ so that:

$$\Pr\left[\begin{array}{c|r} V_D(\mathsf{st}', x, b_1, \dots, b_q) = 1 & (\mathsf{st}', \mathbf{a}_1, \dots, \mathbf{a}_q) \leftarrow V_Q(1^n) \\ & \forall i \in [q],\ b_i = f_i(\mathbf{a}_i) \end{array}\right] > \delta \ ,$$

which contradicts soundness of the strong LMIP.

Knowledge soundness. Let Ext be the extractor of the LMIP. We specify our SNARK extractor Ext':

1. On input $\mathsf{crs} = (\mathsf{pk}, \mathsf{ct}_1, \dots, \mathsf{ct}_\ell)$, x, cct, and a trace tr.

2. Run $(f_1, \ldots, f_q) \leftarrow \mathcal{S}(\mathsf{pk}, \mathsf{cct}, \mathsf{tr})$.
3. Output $w \leftarrow \mathsf{Ext}^{f_1,\ldots,f_q}(x)$.

It is immediate that Ext runs in expected time that is polynomial in λ, n, and t. The proof that our scheme has knowledge soundness $\kappa + \varepsilon_{\mathsf{sem}} + \varepsilon_{\mathsf{ih}}$ closely follows that of standard soundness. Indeed, the knowledge soundness game is:

$$\Pr\left[\begin{array}{c|r} (x,w) \notin \mathcal{R} & \mathcal{G} \leftarrow \mathsf{GGM}(\lambda) \\ \wedge\ \mathbf{V}^{\mathcal{G}}(\mathsf{st}, x, \mathsf{pf}) = 1 & (\mathsf{crs}, \mathsf{st}) \leftarrow \mathsf{Setup}^{\mathcal{G}}(1^n) \\ & (x, \mathsf{pf}) \overset{\mathsf{tr}}{\leftarrow} \mathbf{P}'^{\mathcal{G}}(\mathsf{crs}) \\ & w \leftarrow \mathsf{Ext}(x, \mathsf{pf}, \mathsf{tr}) \end{array}\right]$$

$$= \Pr\left[\begin{array}{c|r} & \mathcal{G} \leftarrow \mathsf{GGM}(\lambda) \\ & (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}} \\ & (\mathsf{st}', \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_Q(1^n) \\ (x,w) \notin \mathcal{R} & \forall i \in [\ell],\ \mathbf{z}_i = (\mathbf{a}_1[i], \ldots, \mathbf{a}_q[i]) \\ \wedge\ \mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct}) \neq \bot & \forall i \in [\ell],\ \mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, \mathbf{z}_i) \\ \wedge\ V_D(\mathsf{st}', x, b_1, \ldots, b_q) = 1 & (\mathsf{cct}, x) \overset{\mathsf{tr}}{\leftarrow} \mathbf{P}'^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell) \\ & (f_1, \ldots, f_q) \leftarrow \mathcal{S}(\mathsf{tr}, \mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell, \mathsf{cct}) \\ & (b_1, \ldots, b_q) \leftarrow \mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct}) \\ & w \leftarrow \mathsf{Ext}^{f_1,\ldots,f_q}(x) \end{array}\right] .$$

By then following precisely the same arguments as in soundness, with this experiment we end up at the fact that there exist x and $f_1, \ldots, f_q$ so that,

$$\Pr\left[\begin{array}{c|r} (x,w) \notin \mathcal{R} & (\mathsf{st}', \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_Q(1^n) \\ \wedge\ V_D(\mathsf{st}', x, b_1, \ldots, b_q) = 1 & \forall i \in [q],\ b_i = f_i(\mathbf{a}_i) \\ & w \leftarrow \mathsf{Ext}^{f_1,\ldots,f_q}(x) \end{array}\right] > \kappa\ ,$$

which contradicts knowledge soundness of the strong LMIP. □

7.2 Construction from bounded-limited homomorphism

In this section we show how to combine a modded LPCP with knowledge soundness, and a bound-limited homomorphic encryption scheme into a designated-verifier SNARK:

Lemma 7.5. *Suppose the existence of the following ingredients:*

- *An input-oblivious mod-LPCP over big modulus p' and moduli (p_i) for a relation $\mathcal{R}$ that is B-bounded with error α, soundness δ, knowledge soundness κ, q queries, query length ℓ, prover running time t_P, and verifier running time (t_Q, t_D). Let B' be its big moduli bound.*

- *A compressible linearly homomorphic encryption scheme over p' with q slots, plaintext moduli (p_i), ciphertext size σ_{ct}, compressed ciphertext size σ_{cct}, decryption bound B, and encryption, compression, evaluation, and decryption times $(t_{\mathsf{enc}}, t_{\mathsf{cmp}}, t_{\mathsf{eval}}, t_{\mathsf{dec}})$. Furthermore, let $\varepsilon_{\mathsf{cor}}$ be its correctness error, $\varepsilon_{\mathsf{bnd}}$ be its bound-limited homomorphism error with bound B' error and $\varepsilon_{\mathsf{sem}}$ be its semantic security advantage.*

Then there is a designated-verifier SNARK for $\mathcal{R}$ with:

- *Completeness error: $\ell \cdot \varepsilon_{\mathsf{cor}}(\lambda)$;*
- *Soundness: $\delta + \varepsilon_{\mathsf{bnd}}(\lambda, t, \ell) + \varepsilon_{\mathsf{sem}}(\lambda, t, \ell)$ against t-query adversaries;*
- *Knowledge soundness: $\kappa + \varepsilon_{\mathsf{bnd}}(\lambda, t, \ell) + \ell \cdot \varepsilon_{\mathsf{sem}}(\lambda, t)$ against t-query adversaries;*
- *Message length: $\sigma_{\mathsf{cct}}(\lambda)$;*
- *CRS length: $\ell \cdot \sigma_{\mathsf{ct}}(\lambda)$;*

- *Setup time:* $O(t_Q + \ell \cdot t_{\mathsf{enc}}(\lambda))$*;*
- *Prover runtime:* $O(t_P + t_{\mathsf{eval}}(\lambda, \ell))$*;*
- *Verifier runtime:* $O(t_D + t_{\mathsf{dec}}(\lambda))$*.*

Proof. Let $(\mathbf{P}, (V_Q, V_D))$ be the linear MIP, and $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ be the homomorphic encryption scheme. We construct a SNARK

Construction 7.6. The SNARK $(\mathsf{Setup}, \mathbf{P}', \mathbf{V}')$ is defined as follows:

- $\mathsf{Setup}^{\mathcal{G}}(1^n)$:
  1. Sample $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}}$ and $(\mathsf{st}', \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_Q(1^n)$.
  2. For every $i \in [\ell]$ let $\mathbf{z}_i = (\mathbf{a}_1[i], \ldots, \mathbf{a}_q[i]) \in \mathbb{F}^q$ and compute $\mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, \mathbf{z}_i)$.
  3. Output $\mathsf{crs} := (\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$ and $\mathsf{st} := (\mathsf{sk}, \mathsf{st}')$.

- $\mathbf{P}'^{\mathcal{G}}(\mathsf{crs}, x, w)$:
  1. Parse $\mathsf{crs} := (\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$ and compute $\pi \leftarrow \mathbf{P}(x, w)$.
  2. Compute $\mathsf{ct}' \leftarrow \mathsf{Eval}^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell, \pi)$.
  3. Let $\mathsf{cct} := \mathsf{Compress}^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}')$.
  4. Output $\mathsf{pf} := \mathsf{cct}$.

- $\mathbf{V}'^{\mathcal{G}}(\mathsf{st}, x, \mathsf{pf})$:
  1. Parse $\mathsf{st} = (\mathsf{sk}, \mathsf{st}')$ and $\mathsf{pf} = \mathsf{cct}$.
  2. If $\mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct}) = \bot$, then output 0. Otherwise, let $(b_1, \ldots, b_q) = \mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct})$.
  3. Output 1 if $V_D(\mathsf{st}', x, b_1, \ldots, b_q) = 1$ and otherwise output 0.

Completeness, and the complexity parameters follow immediately from the construction. We prove soundness and knowledge by first proving soundness, and then explaining the (small) differences when proving knowledge.

Soundness. Fix λ and a prover $\mathbf{P}'$ for the dv-SNARK soundness experiment that makes t queries. We show that the probability that $\mathbf{V}'$ outputs 1 is at most $\delta + \varepsilon_{\mathsf{sem}} + \varepsilon_{\mathsf{bnd}}$. Suppose towards contradiction of the soundness error of the mod-LPCP that $\mathbf{P}'$ causes the verifier to accept with probability $\delta' > \delta + \varepsilon_{\mathsf{sem}} + \varepsilon_{\mathsf{bnd}}$.

Let $\mathcal{S}$ be the simulator of the bounded-restricted homomorphism experiment of the encryption scheme, and define a plaintext generator $\mathcal{T}$ and adversary $\mathcal{A}$:

- $\mathcal{T}^{\mathcal{G}}(\mathsf{pk})$: Sample $(\mathsf{st}', \mathbf{a}_1, \ldots, \mathbf{a}_1) \leftarrow V_Q(x)$. Output $(\mathsf{st}', (\mathbf{z}_1, \ldots, \mathbf{z}_\ell))$ where $\mathbf{z}_i = (\mathbf{a}_1[i], \ldots, \mathbf{a}_q[i])$.
- $\mathcal{A}^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$: Let $\mathsf{crs} = (\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$ and compute and output $(x, \mathsf{cct}) \leftarrow \mathbf{P}'^{\mathcal{G}}(\mathsf{crs})$.

By plugging in to the bound-limited homomorphism experiment this plaintext generator and adversary, the outputs of the following two experiments have statistical distance at most $\varepsilon_{\mathsf{bnd}}$:

- **Real**:
  1. Sample $\mathcal{G} \leftarrow \mathsf{GGM}(\lambda)$.
  2. Let $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}}$.
  3. $(\mathsf{st}', \mathbf{a}_1, \ldots, \mathbf{a}_\ell) \leftarrow V_Q(1^n)$.
  4. $\mathbf{z}_i = (\mathbf{a}_1[i], \ldots, \mathbf{a}_q[i])$ for all $i \in [\ell]$.
  5. $\mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, \mathbf{z}_i)$ for all $i \in [\ell]$.
  6. $(x, \mathsf{cct}) \leftarrow \mathbf{P}'^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$.
  7. If $\mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct}) = \bot$, output $\bot$.
  8. $(b_1, \ldots, b_q) \leftarrow \mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct})$
  9. Output $(\mathsf{st}', x, b_1, \ldots, b_q)$.

- **Ideal**:

1. Let $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}}$.
2. $(\mathsf{st}', \mathbf{a}_1, \ldots, \mathbf{a}_\ell) \leftarrow V_Q(1^n)$.
3. $\mathbf{z}_i = (\mathbf{a}_1[i], \ldots, \mathbf{a}_q[i])$ for all $i \in [\ell]$.
4. $\mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, \mathbf{z}_i)$ for all $i \in [m]$.
5. $(\mathsf{cct}, x) \xleftarrow{\mathsf{tr}} \mathbf{P}'^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$.
6. $(\pi, c_1, \ldots, c_q) \leftarrow \mathcal{S}(\mathsf{tr}, \mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell, \mathsf{cct})$, where $\pi \in \mathbb{Z}_{p'}^q$ and $c_i \in \mathbb{Z}_{p_i}$.
7. If $\mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct}) = \bot$, output $\bot$.
8. $b_i \leftarrow \mathbb{Z}_{p_i}\left(\sum_{j \in [q]} \pi_j \mathbf{z}_j[i]\right) + c_i = \mathbb{Z}_{p_i}(\langle \pi, \mathbf{a}_i \rangle) + c_i$.
9. If there exists an i such that $b_i \notin [-B', B']$ output $\bot$.
10. Output $(\mathsf{st}', x, b_1, \ldots, b_q)$.

Consider the following predicate $p(X)$: if $X = \bot$ or X cannot be parsed as $X = (\mathsf{st}', x, b_1, \ldots, b_q)$, then output 0. Otherwise, output 1 if $|x| = n$, $x \in \mathcal{L}(\mathcal{R})$, and $V_D(\mathsf{st}', x, b_1, \ldots, b_q) = 1$, and otherwise output 0. Observe that after applying the predicate to the output of the real experiment, we get the predicate of whether the SNARK verifier accepted in the adaptive soundness experiment, which happens with probability δ'. Thus, by $\varepsilon_{\mathsf{bnd}}$ statistical indistinguishability of the real and ideal games, the probability of the predicate being satisfied in the ideal game is at least $\delta' - \varepsilon_{\mathsf{bnd}}$:

$$\Pr\left[\begin{array}{c|r} & \mathcal{G} \leftarrow \mathsf{GGM}(\lambda) \\ & (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}} \\ x \notin \mathcal{L}(\mathcal{R}) & (\mathsf{st}', \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_Q(1^n) \\ \wedge\ \mathsf{Dec}^{\mathcal{G}}(\mathsf{sk}, \mathsf{cct}) \neq \bot & \forall i \in [\ell],\ \mathbf{z}_i = (\mathbf{a}_1[i], \ldots, \mathbf{a}_q[i]) \\ \wedge\ \forall i \in [q],\ b_i \in [-B', B'] & \forall i \in [\ell],\ \mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, \mathbf{z}_i) \\ \wedge\ V_D(\mathsf{st}', x, b_1, \ldots, b_q) = 1 & (\mathsf{cct}, x) \xleftarrow{\mathsf{tr}} \mathbf{P}'^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell) \\ & (\pi, c_1, \ldots, c_q) \leftarrow \mathcal{S}(\mathsf{tr}, \mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell, \mathsf{cct}) \\ & \forall i \in [q],\ b_i \leftarrow \mathbb{Z}_{p_i}(\langle \pi, \mathbf{a}_i \rangle) + c_i \end{array}\right] \geq \delta' - \varepsilon_{\mathsf{bnd}}(\lambda, t, \ell) \ .$$

We can now remove the check that the decryption succeeds, thus making the predicate independent of the encryption secret key. That is, the above expression is upper bounded by:

$$\Pr\left[\begin{array}{c|r} & \mathcal{G} \leftarrow \mathsf{GGM}(\lambda) \\ & (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}} \\ & (\mathsf{st}', \mathbf{a}_1, \ldots, \mathbf{a}_q) \leftarrow V_Q(1^n) \\ x \notin \mathcal{L}(\mathcal{R}) & \forall i \in [\ell],\ \mathbf{z}_i = (\mathbf{a}_1[i], \ldots, \mathbf{a}_q[i]) \\ \wedge\ \forall i \in [q],\ b_i \in [-B', B'] & \forall i \in [\ell],\ \mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, \mathbf{z}_i) \\ \wedge\ V_D(\mathsf{st}', x, b_1, \ldots, b_q) = 1 & (\mathsf{cct}, x) \xleftarrow{\mathsf{tr}} \mathbf{P}'^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell) \\ & (\pi, c_1, \ldots, c_q) \leftarrow \mathcal{S}(\mathsf{tr}, \mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell, \mathsf{cct}) \\ & \forall i \in [q],\ b_i \leftarrow \mathbb{Z}_{p_i}(\langle \pi, \mathbf{a}_i \rangle) + c_i \end{array}\right]$$

We now utilize semantic security of the encryption scheme to change the encryptions to 0^q.

$$\Pr\left[\begin{array}{c|r} x \notin \mathcal{L}(\mathcal{R}) & \mathcal{G} \leftarrow \mathsf{GGM}(\lambda) \\ \wedge\, \forall i \in [q],\ b_i \in [-B', B'] & (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}} \\ \wedge\, V_D(\mathsf{st}', x, b_1, \dots, b_q) = 1 & (\mathsf{st}', \mathbf{a}_1, \dots, \mathbf{a}_q) \leftarrow V_Q(1^n) \\ & \forall i \in [\ell],\ \mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, 0^q) \\ & (\mathsf{cct}, x) \xleftarrow{\mathsf{tr}} \mathbf{P}'^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \dots, \mathsf{ct}_\ell) \\ & (\pi, c_1, \dots, c_q) \leftarrow \mathcal{S}(\mathsf{tr}, \mathsf{pk}, \mathsf{ct}_1, \dots, \mathsf{ct}_\ell, \mathsf{cct}) \\ & \forall i \in [q],\ b_i \leftarrow \mathbb{Z}_{p_i}(\langle \pi, \mathbf{a}_i \rangle) + c_i \end{array}\right]$$

$$\geq \Pr\left[\begin{array}{c|r} x \notin \mathcal{L}(\mathcal{R}) & \mathcal{G} \leftarrow \mathsf{GGM}(\lambda) \\ \wedge\, \forall i \in [q],\ b_i \in [-B', B'] & (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}} \\ \wedge\, V_D(\mathsf{st}', x, b_1, \dots, b_q) = 1 & (\mathsf{st}', \mathbf{a}_1, \dots, \mathbf{a}_q) \leftarrow V_Q(1^n) \\ & \forall i \in [\ell],\ \mathbf{z}_i = (\mathbf{a}_1[i], \dots, \mathbf{a}_q[i]) \\ & \forall i \in [\ell],\ \mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, \mathbf{z}_i) \\ & (\mathsf{cct}, x) \xleftarrow{\mathsf{tr}} \mathbf{P}'^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \dots, \mathsf{ct}_\ell) \\ & (\pi, c_1, \dots, c_q) \leftarrow \mathcal{S}(\mathsf{tr}, \mathsf{pk}, \mathsf{ct}_1, \dots, \mathsf{ct}_\ell, \mathsf{cct}) \\ & \forall i \in [q],\ b_i \leftarrow \mathbb{Z}_{p_i}(\langle \pi, \mathbf{a}_i \rangle) + c_i \end{array}\right] - \varepsilon_{\mathsf{sem}}(\lambda, t, \ell) \ .$$

Indeed, otherwise we could run the above experiments to distinguish ciphertexts of $(\mathbf{z}_1, \dots, \mathbf{z}_\ell)$ from ciphertexts of the all-zeroes string. Thus, we have that

$$\Pr\left[\begin{array}{c|r} x \notin \mathcal{L}(\mathcal{R}) & \mathcal{G} \leftarrow \mathsf{GGM}(\lambda) \\ \wedge\, \forall i \in [q],\ b_i \in [-B', B'] & (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}^{\mathcal{G}} \\ \wedge\, V_D(\mathsf{st}', x, b_1, \dots, b_q) = 1 & (\mathsf{st}', \mathbf{a}_1, \dots, \mathbf{a}_q) \leftarrow V_Q(1^n) \\ & \forall i \in [\ell],\ \mathsf{ct}_i \leftarrow \mathsf{Enc}^{\mathcal{G}}(\mathsf{pk}, 0^q) \\ & (\mathsf{cct}, x) \xleftarrow{\mathsf{tr}} \mathbf{P}'^{\mathcal{G}}(\mathsf{pk}, \mathsf{ct}_1, \dots, \mathsf{ct}_\ell) \\ & (\pi, c_1, \dots, c_q) \leftarrow \mathcal{S}(\mathsf{tr}, \mathsf{pk}, \mathsf{ct}_1, \dots, \mathsf{ct}_\ell, \mathsf{cct}) \\ & \forall i \in [q],\ b_i \leftarrow \mathbb{Z}_{p_i}(\langle \pi, \mathbf{a}_i \rangle) + c_i \end{array}\right] \geq \delta' - \varepsilon_{\mathsf{bnd}}(\lambda, \ell, t) - \varepsilon_{\mathsf{sem}}(\lambda, t, \ell) > \delta \ .$$

Observe that, now the choice of x, π, and $c_1, \dots, c_q$ does not depend on $\mathbf{a}_1, \dots, \mathbf{a}_q$. By an averaging argument, there exist x, π, and $c_1, \dots, c_q$ so that:

$$\Pr\left[\begin{array}{c|c} \forall i \in [q],\ b_i \in [-B', B'] & (\mathsf{st}', \mathbf{a}_1, \dots, \mathbf{a}_q) \leftarrow V_Q(1^n) \\ \wedge\, V_D(\mathsf{st}', x, b_1, \dots, b_q) = 1 & \forall i \in [q],\ b_i \leftarrow \mathbb{Z}_{p_i}(\langle \pi, \mathbf{a}_i \rangle) + c_i \end{array}\right] > \delta \ ,$$

which contradicts soundness of the LPCP.

Knowledge soundness. Let Ext be the extractor of the LPCP. We specify our SNARK extractor Ext':

1. On input $\mathsf{crs} = (\mathsf{pk}, \mathsf{ct}_1, \dots, \mathsf{ct}_\ell)$, x, cct, and a trace tr.
2. Run $(\pi, c_1, \dots, c_q) \leftarrow \mathcal{S}(\mathsf{tr}, \mathsf{pk}, \mathsf{ct}_1, \dots, \mathsf{ct}_\ell, \mathsf{cct})$.
3. Output $w \leftarrow \mathsf{Ext}(x, \pi, c_1, \dots, c_q)$.

It is immediate that Ext runs in expected time that is polynomial in λ, n, and t. The proof that our scheme has knowledge soundness $\kappa + \varepsilon_{\mathsf{sem}} + \varepsilon_{\mathsf{bnd}}$ closely follows that of standard soundness. Indeed, the knowledge soundness game is:

$$\Pr\left[\begin{array}{c}(x,w)\notin\mathcal{R}\\ \wedge\ \mathbf{V}^{\mathcal{G}}(\mathsf{st},x,\mathsf{pf})=1\end{array}\ \middle|\ \begin{array}{r}\mathcal{G}\leftarrow\mathsf{GGM}(\lambda)\\ (\mathsf{crs},\mathsf{st})\leftarrow\mathsf{Setup}^{\mathcal{G}}(1^n)\\ (x,\mathsf{pf})\overset{\mathsf{tr}}{\leftarrow}\mathbf{P}'^{\mathcal{G}}(\mathsf{crs})\\ w\leftarrow\mathsf{Ext}(x,\mathsf{pf},\mathsf{tr})\end{array}\right]$$

$$=\Pr\left[\begin{array}{c}(x,w)\notin\mathcal{R}\\ \wedge\ \mathsf{Dec}^{\mathcal{G}}(\mathsf{sk},\mathsf{cct})\neq\bot\\ \wedge\ V_D(\mathsf{st}',x,b_1,\dots,b_q)=1\end{array}\ \middle|\ \begin{array}{r}\mathcal{G}\leftarrow\mathsf{GGM}(\lambda)\\ (\mathsf{pk},\mathsf{sk})\leftarrow\mathsf{KeyGen}^{\mathcal{G}}\\ (\mathsf{st}',\mathbf{a}_1,\dots,\mathbf{a}_q)\leftarrow V_Q(1^n)\\ \forall i\in[\ell],\ \mathbf{z}_i=(\mathbf{a}_1[i],\dots,\mathbf{a}_q[i])\\ \forall i\in[\ell],\ \mathsf{ct}_i\leftarrow\mathsf{Enc}^{\mathcal{G}}(\mathsf{pk},\mathbf{z}_i)\\ (\mathsf{cct},x)\overset{\mathsf{tr}}{\leftarrow}\mathbf{P}'^{\mathcal{G}}(\mathsf{pk},\mathsf{ct}_1,\dots,\mathsf{ct}_\ell)\\ (\pi,c_1,\dots,c_q)\leftarrow\mathcal{S}(\mathsf{tr},\mathsf{pk},\mathsf{ct}_1,\dots,\mathsf{ct}_\ell,\mathsf{cct})\\ (b_1,\dots,b_q)\leftarrow\mathsf{Dec}^{\mathcal{G}}(\mathsf{sk},\mathsf{cct})\\ w\leftarrow\mathsf{Ext}(x,\pi,c_1,\dots,c_q)\end{array}\right].$$

By then following precisely the same arguments as in soundness, with this experiment we end up at the fact that there exist x, π, and $c_1,\dots,c_q$ so that,

$$\Pr\left[\begin{array}{c}(x,w)\notin\mathcal{R}\\ \wedge\ \forall i\in[q],\ b_i\in[-B',B']\\ \wedge\ V_D(\mathsf{st}',x,b_1,\dots,b_q)=1\end{array}\ \middle|\ \begin{array}{r}(\mathsf{st}',\mathbf{a}_1,\dots,\mathbf{a}_q)\leftarrow V_Q(1^n)\\ \forall i\in[q],\ b_i\leftarrow\mathbb{Z}_{p_i}(\langle\pi,\mathbf{a}_i\rangle)+c_i\\ w\leftarrow\mathsf{Ext}(x,\pi,c_1,\dots,c_q)\end{array}\right]>\kappa\ ,$$

which contradicts knowledge soundness of the strong LMIP. □

Acknowledgments

Gal Arnon is supported by the European Research Union (ERC, CRYPTOPROOF, 101164375), and by an Alon Young Faculty Fellowship. Jesko Dujmovic is supported by the European Reasearch Union (ERC, LACONIC, 101041207). Yuval Ishai was supported by ISF grant 3527/24, BSF grant 2022370, and ISF-NSFC grant 3127/23.

References

[ABCH19] Per Austrin, Jonah Brown-Cohen, and Johan Håstad. Optimal inapproximability with universal factor graphs. *ACM Transactions on Algorithms*, 2019.

[ACFY24a] Gal Arnon, Alessandro Chiesa, Giacomo Fenzi, and Eylon Yogev. STIR: Reed-solomon proximity testing with fewer queries. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part X*, volume 14929 of *LNCS*, pages 380–413, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland.

[ACFY24b] Gal Arnon, Alessandro Chiesa, Giacomo Fenzi, and Eylon Yogev. WHIR: reed-solomon proximity testing with super-fast verification. *IACR Cryptol. ePrint Arch.*, page 1586, 2024.

[AFLN24] Martin R. Albrecht, Giacomo Fenzi, Oleksandra Lapiha, and Ngoc Khanh Nguyen. SLAP: Succinct lattice-based polynomial commitments from standard assumptions. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part VII*, volume 14657 of *LNCS*, pages 90–119, Zurich, Switzerland, May 26–30, 2024. Springer, Cham, Switzerland.

[AHRS01] Yonatan Aumann, Johan Håstad, Michael O. Rabin, and Madhu Sudan. Linear-consistency testing. *J. Comput. Syst. Sci.*, 62(4):589–607, 2001.

[BBB$^+$18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Gregory Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 315–334. IEEE Computer Society, 2018.

[BBD$^+$20] Zvika Brakerski, Pedro Branco, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Constant ciphertext-rate non-committing encryption from standard assumptions. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 58–87, Durham, NC, USA, November 16–19, 2020. Springer, Cham, Switzerland.

[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, pages 14:1–14:17, 2018.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, 1988.

[BCC$^+$16] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 327–357, Vienna, Austria, May 8–12, 2016. Springer, Berlin, Heidelberg, Germany.

[BCC$^+$17] Nir Bitansky, Ran Canetti, Alessandro Chiesa, Shafi Goldwasser, Huijia Lin, Aviad Rubinstein, and Eran Tromer. The hunting of the SNARK. *J. Cryptol.*, 30(4):989–1066, 2017.

[BCI$^+$13] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 315–333, Tokyo, Japan, March 3–6, 2013. Springer, Berlin, Heidelberg, Germany.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 31–60, Beijing, China, October 31 – November 3, 2016. Springer, Berlin, Heidelberg, Germany.

[BGI16] Elette Boyle, Niv Gilboa, and Yuval Ishai. Breaking the circuit size barrier for secure computation under DDH. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 509–539, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Berlin, Heidelberg, Germany.

[BGI17] Elette Boyle, Niv Gilboa, and Yuval Ishai. Group-based secure computation: Optimizing rounds, communication, and computation. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 163–193, Paris, France, April 30 – May 4, 2017. Springer, Cham, Switzerland.

[BHI$^+$24] Nir Bitansky, Prahladh Harsha, Yuval Ishai, Ron D. Rothblum, and David J. Wu. Dot-product proofs and their applications. In *65th FOCS*, pages 806–825. IEEE Computer Society Press, October 2024.

[BIOW20] Ohad Barta, Yuval Ishai, Rafail Ostrovsky, and David J. Wu. On succinct arguments and witness encryption from groups. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 776–806, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Cham, Switzerland.

[BISW17] Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu. Lattice-based SNARGs and their application to more efficient obfuscation. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 247–277, Paris, France, April 30 – May 4, 2017. Springer, Cham, Switzerland.

[BISW18] Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu. Quasi-optimal SNARGs via linear multi-prover interactive proofs. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 222–255, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Cham, Switzerland.

[BLR93] Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. *J. Comput. Syst. Sci.*, 47(3):549–595, 1993.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press.

[BS23] Ward Beullens and Gregor Seiler. LaBRADOR: Compact proofs for R1CS from module-SIS. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 518–548, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland.

[CY24] Alessandro Chiesa and Eylon Yogev. Building cryptographic proofs from hash functions. *URL: https://github. com/hash-based-snargs-book*, 2024.

[DFGK14] George Danezis, Cédric Fournet, Jens Groth, and Markulf Kohlweiss. Square span programs with applications to succinct NIZK arguments. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 532–550, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014. Springer, Berlin, Heidelberg, Germany.

[DGI$^+$19] Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. Trapdoor hash functions and their applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 3–32, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Cham, Switzerland.

[DKK18] Itai Dinur, Nathan Keller, and Ohad Klein. An optimal distributed discrete log protocol with applications to homomorphic secret sharing. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 213–242, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Cham, Switzerland.

[DMS24] Michel Dellepere, Pratyush Mishra, and Alireza Shirzad. Garuda and pari: Faster and smaller SNARKs via equifficient polynomial commitments. Cryptology ePrint Archive, Report 2024/1245, 2024.

[FJ12] Uriel Feige and Shlomo Jozeph. Universal factor graphs. In *Automata, Languages, and Programming: 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I 39*, pages 339–350. Springer, 2012.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645, Athens, Greece, May 26–30, 2013. Springer, Berlin, Heidelberg, Germany.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340, Singapore, December 5–9, 2010. Springer, Berlin, Heidelberg, Germany.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326, Vienna, Austria, May 8–12, 2016. Springer, Berlin, Heidelberg, Germany.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108, San Jose, CA, USA, June 6–8, 2011. ACM Press.

[Hås01] Johan Håstad. Some optimal inapproximability results. *Journal of the ACM (JACM)*, 48(4):798–859, 2001.

[HK05] Johan Håstad and Subhash Khot. Query efficient pcps with perfect completeness. *Theory Comput.*, 1(1):119–148, 2005.

[IKO07] Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. Efficient arguments without short pcps. In *22nd Annual IEEE Conference on Computational Complexity (CCC 2007), 13-16 June 2007, San Diego, California, USA*, pages 278–291. IEEE Computer Society, 2007.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *24th ACM STOC*, pages 723–732, Victoria, BC, Canada, May 4–6, 1992. ACM Press.

[Lip13] Helger Lipmaa. Succinct non-interactive zero knowledge arguments from span programs and linear error-correcting codes. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 41–60, Bengalore, India, December 1–5, 2013. Springer, Berlin, Heidelberg, Germany.

[Lip24] Helger Lipmaa. Polymath: Groth16 is not the limit. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part X*, volume 14929 of *LNCS*, pages 170–206, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland.

[LM19] Russell W. F. Lai and Giulio Malavolta. Subvector commitments with application to succinct arguments. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 530–560, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Cham, Switzerland.

[Mic94] Silvio Micali. CS proofs (extended abstracts). In *35th FOCS*, pages 436–453, Santa Fe, NM, USA, November 20–22, 1994. IEEE Computer Society Press.

[MS24] Daniele Micciancio and Mark Schultz-Wu. Bit security: Optimal adversaries, equivalence results, and a toolbox for computational-statistical security analysis. In Elette Boyle and Mohammad Mahmoody, editors, *Theory of Cryptography - 22nd International Conference, TCC 2024, Milan, Italy, December 2-6, 2024, Proceedings, Part II*, volume 15365 of *Lecture Notes in Computer Science*, pages 224–254. Springer, 2024.

[RRR16] Omer Reingold, Ron Rothblum, and Guy Rothblum. Constant-round interactive proofs for delegating computation. In *Proceedings of the 48th ACM Symposium on the Theory of Computing*, STOC '16, pages 49–62, 2016.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266, Konstanz, Germany, May 11–15, 1997. Springer, Berlin, Heidelberg, Germany.

[SSE$^+$24] 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 *ACM CCS 2024*, pages 3167–3181. ACM Press, November 2024.

[SW21] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. *SIAM J. Comput.*, 50(3):857–908, 2021.

A On Measuring Concrete Proof Length

In this section we discuss in more detail our measures of *concrete* (as opposed to asymptotic) proof length, which follow the standards of previous related works, and discuss an important related distinction between public and designated verification.

Concrete proof length is only meaningful when specifying a concrete security level. While there are several principled approaches for defining the exact "bit security" of cryptographic primitives (see, e.g., [MS24] and references therein), the common practice in generic model constructions is to simply refer to the bit-length of the oracle instantiation. For instance, a garbled circuit construction in the ROM is referred to as having 128-bit security when the random oracle is instantiated using (say) AES, even if the security reduction involves some *multiplicative* polynomial loss (as opposed to quadratic loss required by collision resistance). Similarly, in GGM constructions that rely on the hardness of computing discrete logarithms, a 128-bit security level refers to an instantiation with suitable elliptic-curve groups whose order is a 256-bit prime. This accounts for the quadratic speedup of fast algorithms for computing discrete logarithm. We follow this convention here as well.

Finally, we would like to point out an important distinction between dv-SNARGs and publicly verifiable SNARGs in the context of measuring concrete security. In publicly verifiable proofs, the prover can test whether a proof they generate is accepted by the verifier. It is therefore natural to only measure the expected time it takes for a malicious prover to generate an accepted false proof. In contrast, in a designated verifier setting, where the prover does not know whether they will be caught cheating, there is a natural separation between the computational security level and the statistical soundness error.

For example, soundness as low as 2^{-64} is more than enough in most use-cases of dv-SNARGs, when a malicious prover cannot verify their own proofs without access to a verification oracle. Indeed, an access to a verification oracle is typically much more costly than just checking a publicly verifiable proof. Consider an extreme scenario in which it costs \$0.0001 to query a verification oracle, and if a prover manages to cheat, they gain the entire earth's GDP ($\approx \$10^{14}$ in 2022). With soundness error 2^{-64}, even in this extreme scenario a malicious prover who tries to cheat has a negative expected utility.

Given the above, our concrete measures of proof size refer to the arguably conservative setting of 2^{-80} soundness error at a 128-bit security level, where the latter refers to using a 256-bit group and ignores the small loss in the security reduction.