On Succinct Arguments and Witness Encryption from Groups

Ohad Barta[*] Yuval Ishai[†] Rafail Ostrovsky[‡] David J. Wu[§]

Abstract

Succinct non-interactive arguments (SNARGs) enable proofs of NP statements with very low communication. Recently, there has been significant work in both theory and practice on constructing SNARGs with very short proofs. Currently, the state-of-the-art in succinctness is due to Groth (Eurocrypt 2016) who constructed a SNARG from *bilinear* maps where the proof consists of just 3 group elements.

In this work, we first construct a concretely-efficient designated-verifier (preprocessing) SNARG with inverse polynomial soundness, where the proof consists of just 2 group elements in a *standard* (generic) group. This leads to a 50% reduction in concrete proof size compared to Groth's construction. We follow the approach of Bitansky et al. (TCC 2013) who describe a compiler from linear PCPs to SNARGs in the preprocessing model. Our improvement is based on a new *linear PCP packing* technique that allows us to construct 1-query linear PCPs which can then be compiled into a SNARG (using ElGamal encryption over a generic group). An appealing feature of our new SNARG is that the verifier can precompute a *statement-independent* lookup table in an offline phase; verifying proofs then only requires 2 exponentiations and a single table lookup. This makes our new designated-verifier SNARG appealing in settings that demand fast verification and minimal communication.

We then turn to the question of constructing arguments where the proof consists of a *single* group element. Here, we first show that any (possibly interactive) argument for a language $\mathcal{L}$ where the verification algorithm is "generic" (i.e., only performs generic group operations) and the proof consists of a single group element, implies a *witness encryption* scheme for $\mathcal{L}$. We then show that under a yet-unproven, but highly plausible, hypothesis on the hardness of approximating the minimal distance of linear codes, we can construct a 2-message laconic argument for NP where the proof consists of a single group element. Under the same hypothesis, we obtain a witness encryption scheme for NP in the generic group model. Along the way, we show that under a conceptually-similar but *proven* hardness of approximation result, there is a 2-message laconic argument for NP with negligible soundness error where the prover's message consists of just 2 group elements. In both settings, we obtain laconic arguments (and linear PCPs) with *linear* decision procedures. Our constructions circumvent a previous lower bound by Groth on such argument systems with linear decision procedures by relying on *imperfect completeness*. Namely, our constructions have vanishing but not negligible completeness error, while the lower bound of Groth implicitly assumes negligible completeness error of the underlying argument. Our techniques thus highlight new avenues for designing linear PCPs, succinct arguments, and witness encryption schemes.

[*]Technion. Email: ohadba@cs.technion.ac.il. Supported by ERC Project NTSC (742754).

[†]Technion. Email: yuvali@cs.technion.ac.il. Supported by ERC Project NTSC (742754), NSF-BSF grant 2015782, BSF grant 2018393, and a joint Israel-India grant. Part of this work was done while visiting the Simons Institute for the Theory of Computing.

[‡]UCLA. Email: rafail@cs.ucla.edu. Supported in part by DARPA under Cooperative Agreement No. HR0011-20-2-0025, NSF-BSF Grant 1619348, US-Israel BSF grant 2012366, Google Faculty Award, JP Morgan Faculty Award, IBM Faculty Research Award, Xerox Faculty Research Award, OKAWA Foundation Research Award, B. John Garrick Foundation Award, Teradata Research Award, and Lockheed-Martin Corporation Research Award. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA, the Department of Defense, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes not withstanding any copyright annotation therein.

[§]University of Virginia. Email: dwu4@virginia.edu. Supported by NSF CNS-1917414 and a University of Virginia SEAS Research Innovation Award. Part of this work was done while visiting the Simons Institute for the Theory of Computing.

1 Introduction

This work is motivated by two, seemingly unrelated, questions: (1) How *succinct* can practical proof systems be? (2) Can we base *witness encryption* on "20^{th} century cryptography?" We make progress on both questions, which turn out to be related, by introducing new techniques for constructing succinct proof systems in the *generic group model*. We start with relevant background.

Interactive proof systems [GMR85] provide a general framework that allows a verifier to efficiently check claims made by a (possibly malicious) prover. The two properties we require from an interactive proof system are *completeness*, which says that an honest prover should successfully convince an honest verifier of a true statement, and *soundness*, which says that a malicious prover should not be able to convince an honest verifier of a false statement, except perhaps with small probability referred to as *soundness error*.

An important metric in the design of interactive proof systems is the *communication complexity*, and specifically, the amount of communication from the prover to the verifier. For an NP language, an interactive proof system is said to be *laconic* or *succinct* if the total communication from the prover to the verifier is *sublinear* in the size of the NP witness. In the setting of general NP languages, non-trivial savings in the prover-to-verifier communication (beyond sending the classical NP witness) are unlikely if we require the proof system to be statistically sound (i.e., sound even against an unbounded prover) [BHZ87, GH98, GVW01, Wee05b]. If we relax the requirements and only consider proof systems with computational soundness (known as "argument systems" [BCC88]), significant efficiency improvements are possible.

Starting from the seminal work of Kilian [Kil92], who gave the first construction of an interactive laconic argument from probabilistically checkable proofs (PCPs) and collision-resistant hash functions, a large body of work investigated computationally sound proof systems for NP that have low communication complexity, typically polylogarithmic in the size of the classical NP witness. These include interactive *laconic arguments* [GVW01, IKO07], with low prover-to-verifier communication, and *succinct non-interactive arguments* ("SNARGs" [GW11]) that only include a single, short proof message and may involve a trusted setup (cf. [Mic00, Mie08, CL08, Gro10, BCCT12, Lip12, BC12, GGPR13, BCI+13, DFGK14, Gro16, BCC+16, BCC+17, BISW17, BBB+18, BISW18, BBHR19] and the references therein).

Minimizing proof size. A long sequence of works, beginning with the work of Groth [Gro10], has sought to minimize the concrete proof size in SNARGs for NP. Most of these works rely on a (pairing-friendly) bilinear group, where soundness is based either on strong and "unfalsifiable" assumptions or is proved in an idealized "generic group" model [Nec94, Sho97]. Here we will use the simpler generic model formulation.

Groth's initial construction had proofs with 42 (bilinear) group elements; this was later reduced to 39 elements by Lipmaa [Lip12]. In both constructions, the prover's computational complexity was quadratic in the size of the NP verification circuit. Subsequently, using a new characterization of NP based on quadratic span programs, Gennaro et al. [GGPR13] showed how to construct SNARGs where the proof consists of just 7 group elements and where the prover computation is quasi-linear in the size of the verification circuit. Bitansky et al. [BCI+13] introduced a more abstract view of quadratic span programs as implying a "linear PCP," where a verifier can make a small number of inner product queries to a proof vector, and described a general compiler from linear PCPs to SNARGs using a notion called linear-only encryption. A similar compiler was implicit in [GGPR13], and both of these works follow the high-level blueprint introduced in [IKO07, Gro10]. Danezis et al. [DFGK14] subsequently refined quadratic span programs to square span programs and showed how to construct succinct arguments with just 4 bilinear group elements. This line of work culminated with [Gro16], which showed how to construct succinct arguments with just 3 bilinear group elements (just over 1000 bits in existing implementations [SCI20]) and with very efficient verification. These advances in constructing highly succinct arguments with lightweight verification have served as the basis for a number of efficient implementations [PHGR13, BCG+13, BCG+14, BBFR15]. The work of Groth [Gro16] raises the following natural question on the possibility of even shorter group-based proofs:

Can we construct succinct arguments where the proof consists of just one *or* two *group elements?*

Bitansky et al. [BCI+13] previously showed that by instantiating their compiler with a linear PCP built from classical PCPs (e.g., [ALM+98]) and with the ElGamal encryption scheme [ElG84], one can obtain

a *designated-verifier* SNARG in which the proof consists of just two group elements. (Note that in the designated-verifier setting, the verifier possesses a *secret* key that it uses to check proofs [KMO89].) A limitation of the construction from [BCI+13] is the inherent reliance on "classical" PCPs, where the verifier is restricted to read individual symbols of the proof instead of the inner-product queries of a linear PCP. This greatly reduces the concrete efficiency of the resulting construction in comparison to alternative constructions based on linear PCPs.

Groth's general construction [Gro16] can also be applied to linear PCPs based on square-span programs [DFGK14] to obtain a (publicly-verifiable) pairing-based argument where the proof consists of just 2 group elements. The drawback of this particular instantiation of Groth's construction is that it requires a *symmetric* pairing (i.e., a Type I pairing) to implement the verification algorithm. In contrast, the 3-group-element version of Groth based on square span programs or quadratic span programs can be instantiated with an asymmetric pairing (i.e., a Type III pairing). Both the verification cost as well as the overall proof size (in bits) of the 2-group-element construction is higher than those of the 3-group-element construction (due to the larger parameter sizes needed to instantiate a symmetric pairing group at a target security level). For comparison purposes in this work, we focus on the most efficient 3-group-element SNARG of Groth.

1.1 Summary of Contributions

In this work, we develop new techniques for constructing designated-verifier SNARGs[1] and laconic arguments for NP where the proof consists of just two elements or even just one element in a *standard* (rather than bilinear) generic group, at the cost of settling for non-negligible soundness or completeness error. We then apply such a proof system to obtain witness encryption under a new (and unproven) hardness of approximation hypothesis. More concretely, we obtain the following results.

- **Concretely-efficient SNARGs with 2 group elements:** We introduce a new "packing" technique for constructing *1-query* linear PCPs from k-query linear PCPs. We then apply the compiler from [BCI+13], in conjunction with ElGamal encryption,[2] to obtain a designated-verifier SNARG where the proofs consist of two group elements (in a *pairing-free* group). Compared to the pairing-based SNARGs of [Gro16], our arguments are *half* as long (64 bytes vs. 127 bytes), and moreover, with a precomputed verification table, the verification complexity of our SNARG requires only 2 group exponentiations (and 2 multiplications). This is faster (typically by 10x or more) than the verification complexity of [Gro16], which requires 3 pairing operations and multiple exponentiations/multiplications. Compared to [BCI+13], our SNARGs are based on *linear PCPs* rather than classical PCPs, so they also enjoy concretely-efficient prover complexities for small circuits. At the same time, compared to [Gro16], our constructions are in the designated-verifier setting, have a quadratic-size CRS (as opposed to a linear-size CRS), and have inverse polynomial soundness error (as opposed to negligible soundness error). However, the fast verification time and shorter proof size make our construction naturally suited for a number of scenarios (see Section 1.2).
- **Laconic arguments with 2 group elements and negligible soundness error:** The aboove SNARGs, obtained by combining a 1-query linear PCP in conjunction with ElGamal encryption, have inverse polynomial soundness error. This limitation is due to two factors: (1) the linear PCP verification procedure is *non-linear* in the responses (for both the original [BCI+13] proposal based on standard PCPs as well as the linear PCPs obtained via our packing transformation); and (2) decryption in the (additively homomorphic variant of) ElGamal encryption requires computing a discrete log. If however we can construct a 1-query linear PCP with negligible soundness error and where the decision procedure is *linear*, then we can apply the [BCI+13] compiler (with ElGamal) to obtain a 2-element SNARG with negligible soundness error. On the one hand, [Gro16] previously ruled out such a linear

[1] As we discuss in greater detail in Section 1.2, our constructions naturally extend via standard techniques to provide *zero-knowledge* and *arguments of knowledge* (namely, they are "zkSNARKs"). For simplicity of exposition, we just focus on SNARGs here.

[2] Specifically, we rely on the assumption that the ElGamal encryption scheme satisfies linear targeted malleability [BSW12, BCI+13]. We show in Appendix C.1 that this holds in the standard generic group model [Nec94, Sho97].

PCP by showing that soundness can be violated by solving a system of linear equations. However, this previous lower bound only applies if the underlying linear PCP has sufficiently small completeness error (see Remark 4.9). In this work, by relying on hardness of approximation for problems related to linear codes, we obtain a 1-query linear PCP with a linear decision procedure, negligible soundness error, and $o(1)$ (but not negligible) completeness error. The linear PCP we obtain has the property that the verifier's queries depend on the statement, and as such, we do not obtain a SNARG via the [BCI+13] compiler. Instead, we obtain the first group-based laconic argument for NP where the prover's message consists of just 2 group elements and has negligible soundness error (either unconditionally in the generic group model or assuming linear targeted malleability of ElGamal).

- **Laconic arguments with 1 group element:** We then turn to the question of whether we can *further* reduce the prover-to-verifier communication. Here, under a yet-unproven, but highly plausible, hypothesis on the hardness of approximating the minimal distance of linear codes (Hypothesis 5.12), we construct a 2-message laconic argument for NP where the prover's message consists of just a single group element. We note that while there is a linear PCP associated with this language (Remark 5.18), our 1-element laconic argument construction does not follow the [BCI+13] compiler, and it is not clear how to leverage the [BCI+13] compiler to obtain an argument system where the proof is a single group element. Instead, we give a direct construction of a 1-element laconic argument that is provably secure in the generic group model, under the hardness of approximation hypothesis mentioned above.

We summarize our main new constructions of SNARGs and laconic arguments in Table 1 and also compare against existing results.

From laconic arguments to witness encryption. Several works [FNV17, BISW18, BDRV18] have studied the connection between laconic arguments and different types of encryption schemes. Notably, Faonio et al. [FNV17] show that any (even non-laconic) argument of knowledge for a language $\mathcal{L}$ where the verifier can *predict* in advance the prover's message implies an extractable *witness encryption* [GGSW13] scheme for $\mathcal{L}$. As noted in [FNV17], their construction also shows an equivalence between predictable arguments (*without* knowledge) and (non-extractable) witness encryption.

Boneh et al. [BISW18] subsequently showed that any 1-bit argument system is predictable for languages that are hard on average. In this work, we show that a conceptually-similar result holds for argument systems where the proof consists of a *single* group element. In particular, we show that any such argument system that has negligible soundness error, and where the verification algorithm can be implemented by a "generic" algorithm (i.e., it only performs generic group operations on the proof), must also be predictable. By [FNV17], such an argument system for a language $\mathcal{L}$ implies a witness encryption scheme for $\mathcal{L}$.

As noted above, if our hypothesis on the hardness of approximation for the minimal distance of linear codes holds, then we obtain a laconic argument for NP with negligible soundness error and where the proof consists of a single group element in the generic group model. Appealing now to the results above, this implies a witness encryption scheme for NP in the generic group model. We stress that, in the generic group model, this result does *not* rely on any cryptographic assumptions; it only relies on a plausible hardness of approximation result that may be *unconditionally* proved in the future. Indeed, there are no known barriers for strengthening the current hardness results to this more demanding parameter regime [Kho20]. Existing constructions of witness encryption all rely on indistinguishability obfuscation [GGH+13], multilinear maps [GGSW13, GLW14, CVW18], or new and yet unexplored algebraic structures [BIJ+20]; thus, a construction in the generic group model would be considered a major development in this area.

Another intriguing implication of this result is that it effectively rules out negative results for constructing witness encryption unconditionally in the generic group model. Such negative results (or barriers) are not only known for powerful primitives such as indistinguishability obfuscation [MMN+16a, MMN+16b], but also for conceptually-simpler primitives such as identity-based encryption [PRV12]. Note that even though identity-based encryption can be built from witness encryption for NP (together with a unique signature scheme) [GGSW13], the resulting construction makes non-black-box use of the group. Thus, a construction of witness encryption in the generic group model does not conflict with existing lower bounds. Indeed,

	Group Type	Number of Elements	Completeness Error	Soundness Error	Proof Type	Verifier Time	PCP vs. LPCP
[Gro16]	bilinear	$2\mathbb{G}_1, 1\mathbb{G}_2$	0	negl	SNARG	$O(1)$	LPCP
[BCI+13]	linear	8	0	1/poly	dvSNARG	$O_\varepsilon(s)$	LPCP
[BCI+13]	linear	2	0	1/poly	dvSNARG	$O_\varepsilon(1)$	PCP
Cor. 3.19	linear	2	0	1/poly	dvSNARG	$O_\varepsilon(s)$	LPCP
Cor. 3.20	linear	2	negl	1/poly	dvSNARG	$O_\varepsilon(\sqrt{s})$	LPCP
Cor. 3.21	linear	2	negl	1/poly	dvSNARG	$O_\varepsilon(1)^*$	LPCP
Cor. 4.7	linear	2	$o(1)$	negl	LA	$O(1)$	PCP
Cor. 5.16†	linear	1	$o(1)$	negl	LA	$O(1)$	PCP

*Using reusable statement-independent prepossessing with $O_\varepsilon(\sqrt{s})$ bits of storage.
†This is a conditional result that relies on a plausible (but yet unproven) hypothesis about hardness of approximation of minimal distance of codes (Hypothesis 5.12).

Table 1: Comparison of our group-based arguments to previous related results. In the "Proof Type" column, SNARG and dvSNARG refer to publicly-verifiable and designated-verifier SNARGs, respectively, and LA refers to 2-message laconic arguments where the verifier's initial message depends on the statement being proved. Verifier time counts group operations as a function of the size s of the classical NP verifier, ignoring polylogarithmic factors, and excluding quasilinear-time preprocessing of the input. An ε-subscript treats the soundness error ε as constant. In the last column, LPCP refers to proof systems obtained from any *linear* PCP whereas PCP refers to proof systems that are based on classical PCPs. The latter do not enjoy reusable soundness and have a very high concrete cost.

an impossibility result for constructing witness encryption in the generic group model would falsify our hypothesis.

Assumptions vs. structures. We would like to stress that the goal of realizing witness encryption in the generic group model is very different from the goal of basing witness encryption on "standard" and well-studied cryptographic assumptions. The latter follows from the very recent indistinguishability obfuscation construction of Jain et al. [JLS20]. Several other constructions of indistinguishability obfuscation based on simple-to-state assumptions, mostly related to the learning with errors (LWE) problem [Reg05], were recently proposed [AP20, GJLS20, BDGM20a, GP20, BDGM20b, WW20]. However, all these recent works employ very different kinds of computational structures than the group structure we use, and additionally make a heavy "non-black-box" use of the underlying cryptographic building blocks.

While we are mainly concerned with the kind of *structure* that suffices for witness encryption, rather than the concrete *assumptions* related to this structure, it seems likely that the generic group in our construction can be replaced by the same kind of standard-model assumptions that were used in the context of SNARGs. This includes extractability assumptions [Mie08, BCCT12, Gro10], "linear-only" assumptions [BCI+13], or the algebraic group model [FKL18]. Given the unproven hardness of approximation hypothesis that underlies our current construction, we view this as a secondary goal.

Finally, it is instructive to draw an analogy between our goal and the celebrated oracle separation result of Impagliazzo and Rudich [IR89]. Impagliazzo and Rudich ask whether "cryptomania" can be reduced to a simple structure used for "minicrypt," where the former is represented by the goal of public-key encryption and the latter by a random oracle. We ask whether (the outskirts of) "obfustopia" can be reduced to a simple structure used for "cryptomania," where the former is represented by the goal of witness encryption and the latter by a random group oracle. Whereas Impagliazzo and Rudich gave a negative answer to their question, we give evidence that the answer to our question may be positive.

1.2 Concretely-Efficient SNARGs with 2 Group Elements

In this section, we provide an overview of our concretely-efficient SNARGs where the proof consists of 2 group elements. Our starting point in this work is the compiler from [BCI+13] (also implicit in [GGPR13]) that compiles a linear PCP into a SNARG in the preprocessing model using a "linear-only" encryption scheme (i.e., an additively-homomorphic encryption scheme that only supports affine operations on ciphertexts).[3] Here, the preprocessing model refers to a SNARG where the running time of the setup algorithm is allowed to depend polynomially in the size of the classical NP verifier. We begin with a brief overview of this compiler.

Linear PCPs. A linear PCP for an NP language $\mathcal{L}$ over a finite field $\mathbb{F}$ is defined by a linear oracle $\boldsymbol{\pi} : \mathbb{F}^\ell \to \mathbb{F}$. On a query $\mathbf{q} \in \mathbb{F}^\ell$, the linear PCP oracle responds with the inner product $\mathbf{q}^\mathsf{T}\boldsymbol{\pi}$. More generally, we can view the linear PCP queries as the columns of a query matrix $\mathbf{Q} \in \mathbb{F}^{\ell \times k}$ and the oracle's operation as computing $\mathbf{Q}^\mathsf{T}\boldsymbol{\pi}$. To verify a proof of a statement $\mathbf{x}$, the verifier submits a query matrix $\mathbf{Q}$ to the oracle and receives back a set of responses $\mathbf{Q}^\mathsf{T}\boldsymbol{\pi}$. In this case, k denotes the number of linear PCP queries the verifier makes. For the language of (Boolean or arithmetic) circuit satisfiability, there exist efficient 3-query linear PCPs based on the quadratic span programs of [GGPR13] with query length $\ell = O(s)$, where s is the size of the circuit. A 2-query variant based on square span programs was given in [DFGK14]. We construct a 2-query linear PCPs based on the Walsh-Hadamard code (see Appendix B) where $\ell = O(s^2)$. This construction improves on the 3-query construction from [ALM+98, IKO07] and, unlike the 2-query construction from [DFGK14], has the feature that the queries can be generated by quadratic (degree-2) polynomials. This feature turns out to be useful towards the goal of efficient 1-query linear PCP.

The Bitansky et al. compiler. A general "cryptographic compiler" of Bitansky et al. [BCI+13] takes any linear PCP and a linear-only encryption scheme and outputs a preprocessing SNARG. If the linear PCP satisfies additional properties such as zero knowledge or knowledge soundness, then the resulting SNARG also inherits those properties (i.e., we can obtain a "zkSNARK"). The idea behind the [BCI+13] compiler is the following: first, they compile a linear PCP into a two-message linear interactive proof (LIP) by introducing an additional consistency check. In this model, the prover is allowed to compute any *affine* function of the verifier's queries (the linear PCP model is more constrained in the sense that the prover has to apply the *same* linear function to each of the verifier's queries). To go from a LIP to a preprocessing SNARG, the verifier encrypts its queries using a linear-only encryption scheme and publishes the ciphertexts as part of the common reference string (CRS). To construct a proof, the prover takes its statement and witness, computes the linear function $\boldsymbol{\pi}$, and homomorphically evaluates $\boldsymbol{\pi}$ on the encrypted queries (this is possible since the honest prover's strategy is linear). The proof is the encrypted set of responses. In the designated-verifier model, the verifier decrypts the responses and applies the standard LIP verification procedure (if the verifier's decision procedure is quadratic, a pairing can be used to perform the verification check "in the exponent," yielding a publicly-verifiable SNARG). Overall, the [BCI+13] compiler takes any k-query linear PCP and compiles it into a preprocessing SNARG where the proofs consist of $(k+1)$ ciphertexts of the underlying linear-only encryption scheme. Under the assumption that the classical ElGamal encryption scheme [ElG84] is linear-only (when the message is encrypted in the exponent), this framework can be used to obtain a SNARG where the proof size consists of $(k+1)$ ElGamal ciphertexts, or equivalently, $2(k+1)$ group elements.

1-query linear PCPs. First, we note that any 1-query linear PCP is itself a 2-message linear interactive proof, and hence, can be directly compiled into a preprocessing SNARG via the [BCI+13] compiler where the proof consists of just a *single* ciphertext (i.e., 2 group elements in the case of ElGamal). However, as noted above, efficient instantiations of linear PCPs based on the Hadamard PCP [ALM+98, IKO07], quadratic span programs [GGPR13] or square span programs [DFGK14, Gro16] all require *at least 2 queries*, and

[3] Technically, a weaker property called linear targeted malleability [BSW12] suffices for a basic version of the compiler. For ease of exposition, we present everything here using the concept of linear-only encryption. We formally define linear targeted malleability in Definition A.6.

thus, cannot be directly compiled into a preprocessing SNARG with 2 group elements. If we start instead from a classical PCP (rather than a linear PCP), then [BCI+13] shows how to construct a 1-query linear PCP, which in conjunction with ElGamal encryption, yields a SNARG with 2-group elements (and inverse polynomial soundness error). However, the use of classical PCPs in this construction incurs a high *concrete* cost, despite significant optimization efforts [BBC+17], even for small verification circuits. As a result, the concrete efficiency of the resulting SNARG is not competitive with existing pairing-based constructions based on efficient linear PCPs. Furthermore, the low entropy of the queries in the PCP-based construction from [BCI+13] prevents the scheme from achieving reusable soundness.[4] In this work, we introduce a new approach to constructing 1-query linear PCPs *without* relying on traditional PCPs. The resulting 1-query linear PCP has reusable soundness (Remark 3.17).

Linear PCP packing. Our first result in this work is a method to *pack* a k-query *linear* PCP into a 1-query linear PCP. Our packing construction is most naturally viewed by considering a linear PCP *over the integers*.[5] Namely, consider a linear PCP where both the query matrix $\mathbf{Q} \in \mathbb{Z}^{\ell \times k}$ and the proof $\boldsymbol{\pi} \in \mathbb{Z}^{\ell}$ consist of vectors over the integers. Clearly, any linear PCP over a finite field $\mathbb{F}_p$ yields a linear PCP over the integers $\mathbb{Z}$ by having the verifier reduce the responses modulo p. We now say that a linear PCP is B-bounded if for every honestly-generated query matrix $\mathbf{Q} \in \mathbb{Z}^{\ell \times k}$ and proof vector $\boldsymbol{\pi} \in \mathbb{Z}^{\ell}$ we have $\|\mathbf{Q}^{\mathsf{T}}\boldsymbol{\pi}\|_{\infty} < B$ (i.e., the magnitude of every response is less than B). Let $\mathbf{q}_1, \ldots, \mathbf{q}_k \in \mathbb{Z}^{\ell}$ be the individual queries (i.e., the columns of $\mathbf{Q}$). Consider the vector $\mathbf{q}_{\text{packed}} = \sum_{i \in [k]} B^{i-1}\mathbf{q}_i \in \mathbb{Z}^{\ell}$. Then,

$$a = \mathbf{q}_{\text{packed}}^{\mathsf{T}}\boldsymbol{\pi} = \sum_{i \in [k]} B^{i-1}\mathbf{q}_i^{\mathsf{T}}\boldsymbol{\pi} \in \mathbb{Z}.$$

If $|\mathbf{q}_i^{\mathsf{T}}\boldsymbol{\pi}| < B$, then a represents an integer in base B where the i^{th} digit is the i^{th} response $\mathbf{q}_i^{\mathsf{T}}\boldsymbol{\pi}$. Thus, by making a single query $\mathbf{q}_{\text{packed}}$ (with much *larger* coefficients), the verifier is able to decode all k responses and implement the verification procedure for the underlying linear PCP. As described, it is not clear that the above approach is sound: namely, an adversary can choose a malicious proof vector $\boldsymbol{\pi}$ such that $\mathbf{Q}^{\mathsf{T}}\boldsymbol{\pi}$ is not B-bounded: then, the tuple of responses decoded using the above procedure would yield a tuple that is not consistent with applying a single consistent linear strategy to all of the query vectors. We solve this problem by *randomizing* the query-packing procedure, similarly to the packing compiler from [BCI+13] for classical PCPs. Namely, instead of using a fixed scaling factor B, the verifier sets $r_1 = 1$ and samples $r_2, \ldots, r_k$ from a sufficiently-large interval and computes the packed query vector as $\mathbf{q}_{\text{packed}} = \sum_{i \in [k]} \mathbf{q}_i \prod_{j \leq i} r_j$. We can now argue that over the verifier's randomness, any adversarial strategy that exceeds the bound will cause the verifier to reject with high probability. We give the construction and analysis in Section 3.

We have now shown how to pack a k-query linear PCP over the integers to obtain a 1-query linear PCP over the integers. To apply the [BCI+13] compiler, we require a linear PCP over a finite field $\mathbb{F}$. Here, we note that we can directly embed the operations over the integers into a sufficiently large finite field (e.g., if B_{packed} is a bound on $\mathbf{q}_{\text{packed}}^{\mathsf{T}}\boldsymbol{\pi}$, it suffices to work over a field $\mathbb{F}_p$ where $p > 2B_{\text{packed}}$). If we start with a linear PCP over $\mathbb{F}_p$ and desire a packed linear PCP over the *same* field $\mathbb{F}_p$, then the linear PCP responses should be small.[6] We refer to the resulting linear PCP as a "bounded" linear PCP over $\mathbb{F}_p$. The Hadamard linear PCP has this property (see Appendix B), so using our basic query-packing transformation, we obtain a 1-query bounded linear PCP over $\mathbb{F}_p$ with query length $\ell = O(s^2)$, where s is the size of the NP verification circuit. A natural question is whether we can obtain a 1-query linear PCP with query length $O(s)$ starting from the quadratic span programs of [GGPR13]. As we explain in Remark 3.13, we are not able to leverage our packing transformation because the queries in those constructions have large coefficients, and thus, do not seem directly amenable to our packing approach.

[4] Indeed, by flipping one bit of an honestly-generated PCP, a malicious prover can mount a selective failure attack that makes the verifier reject with high probability if this bit is being queried.

[5] While we present the general ideas using linear PCPs over the integers, the construction in Section 3 embeds the integer operations over a large finite field $\mathbb{F}_p$.

[6] If the packing transformation requires the use of a larger field than that of the underlying linear PCP, this can negate the benefit of the packing.

Concretely-efficient 2-element SNARGs. Starting from our 1-query linear PCP above, we directly invoke the [BCI+13] compiler with ElGamal encryption to obtain a designated-verifier SNARG in the preprocessing model where the proof consists of 2 group elements. One caveat with ElGamal is that the scheme encodes the message in the *exponent* (i.e., the decryption algorithm recovers g^a rather than a). In the context of the [BCI+13] compiler, this means the linear PCP response is in the exponent, and the verifier has to compute the discrete logarithm in order to verify the proofs; if the size of the response is B-bounded, this can be done in time $\widetilde{O}(\sqrt{B})$ using Pollard's kangaroo algorithm [Pol78]. For this to be efficient, we thus require that the responses are in a polynomial-size interval. Of course, this means that the soundness error achievable using the ElGamal instantiation will be inverse polynomial in the security parameter (rather than negligible). This is because there are now only polynomially-many possible values that causes the verifier to accept, so a malicious prover can guess an accepting value with $1/\mathsf{poly}$ probability. This yields a trade-off between the soundness error ε and the verifier's time complexity (namely, smaller soundness error means that the responses have to be drawn from a larger interval, which increases the running time of the discrete log algorithm). Thus, when compiling linear PCPs to SNARGs using ElGamal, it is natural to consider bounded linear PCPs, which provide a direct trade-off between soundness error and the bound (see Corollary 3.14).

The bound on our 1-query linear PCP based on the Hadamard construction scales with $O(s^4)$, which means the resulting ElGamal-based SNARG will have verification complexity that scales *quadratically* with the circuit size. This is both undesirable and impractical for real scenarios. However, by taking advantage of the structure of the Hadamard linear PCP, we can reduce the verification complexity to $\widetilde{O}(\sqrt{s}/\varepsilon)$ if we allow for a negligible completeness error (as opposed to perfect completeness). The high-level idea here is that in the Hadamard linear PCP (Appendix B.1), one of the (unpacked) query responses is small and lies in an interval of size $\widetilde{O}(\sqrt{s}/\varepsilon)$ with overwhelming probability. This means that instead of having the verifier solve the discrete log to obtain the full linear PCP response, the verifier can instead check whether the decrypted response corresponds to one of the (polynomially-many) accepting values of the Hadamard linear PCP. Thus, we obtain a designated-verifier SNARG with $1/\mathsf{poly}$ soundness error where the proof consists of exactly 2 group elements and the verifier runs in time $\widetilde{O}(\sqrt{s}/\varepsilon)$. We provide the details in Section 3.2.

Preprocessing to achieve constant running time. Our approach for reducing the verification time in the ElGamal-based SNARG described above relies on there only being a small number of accepting values (that depend on the statement and the verifier's secret key). In Section 3.2, we show that at setup time, the verifier can perform a *statement-independent* preprocessing step (which only depends on the verifier's secret verification state) and prepare a lookup table of size $\widetilde{O}(\sqrt{s}/\varepsilon)$. With this lookup table, the verification procedure reduces to performing 2 exponentiations and 2 group multiplications, followed by a single table lookup. This yields a *much faster* verification procedure compared to even the SNARG from [Gro16], which requires computing 3 pairing operations (in addition to multiple exponentiations and group multiplications). In this model, we obtain SNARGs that are both *50% shorter* than those from [Gro16] (64 bytes for our construction vs. 127 bytes for [Gro16, SCI20]) and significantly faster to verify. Based on timings provided in `libsnark` [SCI20], the verifier's running time in [Gro16] is 1.2ms, while based on our estimates, two group exponentiations and two multiplications would take 0.1ms, which is over 10x faster (see Section 3.3 for details on our performance estimates). This makes our designated-verifier SNARGs well-suited for environments that demand very succinct proofs and low-latency or low-energy verification.

Concrete efficiency estimates. In Table 2, we provide concrete estimates on the size of the CRS, the prover complexity, and the verifier complexity. With preprocessing, the primary cost for the verifier is the storage of the lookup table and without preprocessing, the primary cost is the verification time. Here, we apply the additional (standard) transformation to obtain a zkSNARK (described in Section 3.2 and Remark 3.22). We describe our methodology for computing these estimates in Section 3.3.

The main appeal of our new designated-verifier zkSNARKs is that with preprocessing, it has extremely lightweight verification. The proofs consist of just two group elements and with a modestly-sized lookup table (e.g., for circuits with over 15,000 wires and soundness error 1/128, a lookup table of size just over 20 MB suffices). Our schemes are well suited in scenarios where the verifier has a modest amount of memory,

but is otherwise low energy or computationally constrained. They are also well-suited in settings where the verifier might be receiving and authenticating requests from a large number of provers.

One appealing application is to combine the zkSNARK with a one-way function to construct an identification scheme. Here, a user's secret key is a random element in the domain of a one-way function and the public key is its image under the one-way function. To authenticate, the user would provide a zkSNARK proving knowledge of their secret key (i.e., the pre-image under the one-way function) associated with their public key. One way to instantiate the required one-way function is to use Goldreich's simple one-way function based on expander graphs [Gol00], which can be computed by a Boolean circuit with just 1200 gates [CDM+18, BIJ+20] (or 1500 wires). In this case, the CRS size is around 34 MB and the prover's computation would take just a few seconds of computation. With a moderate soundness error of 1/128, the verifier only needs to maintain a table with just over 6 MB of storage. If the bottleneck in the system is sending proofs and authenticating credentials, then our construction offers a compelling solution. Moreover, the expressive nature of zkSNARKs lends itself naturally towards implementing more complex authentication policies (e.g., the user's credential is valid and moreover, satisfies some simple Boolean predicate).

While our construction achieves a lower level of soundness compared to pairing-based alternatives, scenarios where there are severe out-of-band consequences for getting caught cheating (even once) can provide strong incentives for honest behavior. This is conceptually similar to the notion of covert security in multiparty computation [CO99, AL07]. Similarly, while our constructions do not provide perfect zero-knowledge, the effects of any potential leakage can be mitigated (in the above setting with an identification scheme) by using a leakage-resilient one-way function. Moreover, in the setting of short-lived tokens or credentials, the user can simply *refresh* their credential after a certain number of requests (based on the zero-knowledge parameter of the system).

More broadly, we believe that our new preprocessing zkSNARKs are appealing in terms of proof size and verifier complexity. It is interesting to further optimize our methods to support more complex circuits. In Remark 3.16, we describe one approach based on constructing specially-designed circuits that are "Hadamard-friendly," which can then be efficiently-checked using a linear PCP (with small amortized query size), and correspondingly, enable a more concretely efficient zkSNARK.

1.3 From Hardness of Approximation to Witness Encryption

A limitation of the SNARG constructions based on instantiating the [BCI+13] compiler with ElGamal is that they only provide $1/\mathsf{poly}$ soundness error. Part of this stems from the inherent challenge that recovering the linear PCP responses from an ElGamal ciphertext requires computing discrete log, which restricts us to linear PCPs whose responses lie in a polynomial-size set (and correspondingly, yields SNARGs with inverse polynomial soundness error). However, Bitansky et al. [BCI+13] point out that if we had a linear PCP with a *linear* decision procedure and if we apply the compiler using ElGamal encryption, the verifier no longer needs to decrypt the responses. Instead, it can simply check the verification procedure "in the exponent." This provides a general template for constructing a succinct argument based on ElGamal that can achieve negligible soundness error. While Bitansky et al. motivate the search for a linear PCP with a linear decision procedure, they do not suggest a candidate.

1-query linear PCP with negligible soundness error from hardness of approximation. In this work, we introduce a new approach for constructing linear PCPs based on the hardness of approximating problems related to decoding linear codes. Specifically, we construct a 1-query linear PCP with a linear decision procedure and negligible soundness error. Our construction *affirmatively* answers the above question posed by Bitansky et al. on whether there exists a linear interactive proof with a linear decision procedure. We note, however, that the linear PCP we construct is *instance-dependent* (i.e., the verifier's query depends on the statement being verified). As such, applying the [BCI+13] compiler yields a 2-message laconic argument where the prover's message consists of 2 group elements.

Previously, Groth [Gro16] ruled out the possibility of 2-message linear interactive proofs with a linear decision procedure for languages that are hard on average. Implicit in his lower bound is the assumption

Circuit Size	CRS Size	Prover Time	Soundness Error	Verifier Space (with Preproc.)	Verifier Time (without Preproc.)
2^{10}	16MB	262K (3.6s)	2^{-1} 2^{-7} 2^{-14}	58KB 5.3MB 923.4MB	23K(0.33s) 1.5M (21.28s) 194M (45m21s)
2^{12}	256MB	4.2M (58s)	2^{-1} 2^{-7} 2^{-14}	126KB 11.2MB 1.9GB	47K (0.66s) 3M (42.57s) 389M (1h30m)
2^{14}	4GB	67M (15m40s)	2^{-1} 2^{-7} 2^{-14}	270KB 23.5MB 3.9GB	95K (1.33s) 6M (1m25s) 778M (3h1m)

Table 2: Concrete efficiency estimates for our designated-verifier zkSNARK based on ElGamal (see Section 3.2 and Remark 3.22 for details on how to extend the basic SNARG to a zkSNARK). For different circuits sizes (number of wires in a Boolean circuits with fan-in 2 gates) and soundness levels, we measure the CRS size (in group elements), the prover time complexity (in number of group operations), and the verifier complexity (with preprocessing, this corresponds to the size of the lookup table and without preprocessing, this corresponds to the number of group operations needed for online verification). The proof size for *all* of the parameter settings consists of just *two group elements* (64 bytes), and with preprocessing, the verification cost is just 2 exponentiations (and 2 multiplications). We set the completeness error to 2^{-40} and the zero-knowledge parameter to achieve 0.1-statistical zero knowledge. Without zero knowledge, we can reduce the size of the verifier's lookup table or the verifier time (when there is no preprocessing) by 8x. For the concrete timing estimates, we base them on measurements taken using the `libsodium` implementation of the Curve25519 elliptic curve [Ber06] (see Section 3.3 for further details).

that the underlying proof satisfies perfect completeness (or more generally, has negligible completeness error). Our 1-query linear PCP construction has a small, but noticeable, $o(1)$ completeness error which avoids this impossibility. We discuss this in greater detail in Remark 4.9.

Our linear PCP construction relies on the hardness of approximation for the gap minimum weight solution problem (GapMWSP) [KPV12]. At a high level, a problem instance for $\mathsf{GapMWSP}_\beta$ is a triple $(\mathbf{A}, \mathbf{b}, d)$ where $\mathbf{A} \in \mathbb{F}^{\ell \times n}$, $\mathbf{b} \in \mathbb{F}^\ell$, $d \in \mathbb{N}$, and $\mathbb{F}$ is a finite field. The goal is to decide between the following two cases:

- YES instance: there is a solution $\mathbf{x} \in \mathbb{F}^n$ to the linear system $\mathbf{Ax} = \mathbf{b}$ with Hamming weight at most d;
- NO instance: all solutions $\mathbf{x} \in \mathbb{F}^n$ to the linear system $\mathbf{Ax} = \mathbf{b}$ have Hamming weight at least $\beta \cdot d$.

The gap β is referred to as the approximation factor. We will rely on NP-hardness[7] of the above promise problem for some $\beta = \beta(n) \geq \mathsf{polylog}(n)$. While this problem is traditionally formulated over the binary field $\mathbb{F}_2$, we show that the same NP-hardness reduction (from the GapLabelCover problem [Raz95]) extends to general finite fields (Lemma 4.2).

We can construct a linear PCP for the GapMWSP problem in a straightforward manner (this in turn yields a linear PCP for NP by first applying a Karp-Levin reduction to GapMWSP). The linear PCP query for an instance $(\mathbf{A}, \mathbf{b}, d)$ consists of a random vector $\mathbf{r} \xleftarrow{R} \mathbb{F}^\ell$ and a *sparse* vector $\mathbf{e} \in \mathbb{F}^n$ where each component of $\mathbf{e}$ is either uniform over $\mathbb{F}$ or 0. The query is the vector $\mathbf{q}^\mathsf{T} = \mathbf{r}^\mathsf{T}\mathbf{A} + \mathbf{e}^\mathsf{T} \in \mathbb{F}^n$. The proof for an instance $(\mathbf{A}, \mathbf{b}, d)$ is a vector $\boldsymbol{\pi} \in \mathbb{F}^n$ where $\mathbf{A}\boldsymbol{\pi} = \mathbf{b}$ and $\boldsymbol{\pi}$ has small Hamming weight $\mathsf{wt}(\boldsymbol{\pi}) \leq d$. Finally, given a response $a \in \mathbb{F}$, the verifier simply checks whether $a = \mathbf{r}^\mathsf{T}\mathbf{b}$. Suppose that $\mathbf{A}\boldsymbol{\pi} = \mathbf{b}$. Then, $\mathbf{q}^\mathsf{T}\boldsymbol{\pi} = \mathbf{r}^\mathsf{T}\mathbf{A}\boldsymbol{\pi} + \mathbf{e}^\mathsf{T}\boldsymbol{\pi} = \mathbf{r}^\mathsf{T}\mathbf{b} + \mathbf{e}^\mathsf{T}\boldsymbol{\pi}$. Completeness follows as long as $\mathbf{e}^\mathsf{T}\boldsymbol{\pi} = 0$. This happens with $1 - o(1)$ probability since both $\mathbf{e}$ and $\boldsymbol{\pi}$ are *sparse* (i.e., $\mathbf{e}^\mathsf{T}\boldsymbol{\pi}$ is nonzero only if *both* $\mathbf{e}$ and $\boldsymbol{\pi}$ have a nonzero component in the same coordinate, which

[7] Here and in the following, we refer to NP-hardness of *relations* with respect to Karp-Levin reductions. See Section 2 for a formal definition.

happens with small, but noticeable, probability over the randomness of $\mathbf{e}$). Conversely, for a NO instance, all solutions to the linear system $\mathbf{Ax} = \mathbf{b}$ have Hamming weight at least βd. In this case, for any proof vector $\boldsymbol{\pi}$, either $\mathbf{A}\boldsymbol{\pi} \neq \mathbf{b}$ (in which case, the verifier rejects except with probability $1/|\mathbb{F}|$ over the randomness of $\mathbf{r}$) or if $\mathbf{A}\boldsymbol{\pi} = \mathbf{b}$, then $\boldsymbol{\pi}$ has large Hamming weight and $\mathbf{e}^{\mathsf{T}}\boldsymbol{\pi}$ will be nonzero with overwhelming probability over the choice of $\mathbf{e}$. Hence, we obtain an instance-dependent 1-query linear PCP with a linear decision procedure from the GapMWSP problem, and correspondingly, a 2-message laconic argument with negligible soundness error and where the proof size consist of just 2 group elements by invoking the [BCI+13] compiler with ElGamal encryption. We provide the full description and analysis in Section 4.

From laconic arguments to witness encryption. Given a laconic argument where the proof consists of just two group elements, a natural question to ask is whether we can have an argument that is *even shorter*: namely, a laconic argument with just a *single* group element. From a conceptual perspective, this question has a similar flavor to the notion of a "1-bit SNARG" introduced in [BISW18]. There, they showed that a 1-bit SNARG for a hard language is in fact "predictable" (i.e., the verifier can predict the value of an accepting proof), and by leveraging the result from [FNV17], implies a *witness encryption*[8] scheme for the underlying language. As it turns out, laconic arguments where the prover's message is a single group element and where the verification algorithm only consists of *generic group* operations are similarly powerful. As we show in Corollary 5.10, *any* 1-element laconic argument that has negligible soundness error and a "generic" verification algorithm (i.e., it only performs algebraic operations over group elements) implies witness encryption for the underlying language. This means that improving our 2-element laconic argument to a 1-element laconic argument provides a promising new path towards realizing witness encryption from more traditional and well-understood cryptographic assumptions. We provide the details in Section 5.1.

1-element laconic argument from hardness of approximation. It is not clear how to leverage our 1-query linear PCP from the GapMWSP problem to obtain a laconic argument where the proof consists of just a single group element. Indeed, any application of the [BCI+13] compiler with ElGamal would yield an argument system where the proof consists of at least 2 group elements (since the proof will contain at least one ElGamal ciphertext). However, we show that assuming a conceptually-similar, but yet-unproven hypothesis on the hardness of approximating the minimal distance of linear codes, we can leverage similar ideas used to construct our linear PCP from GapMWSP to *directly* construct a 1-element laconic argument with negligible soundness error in the generic group model. The resulting argument is predictable in the sense of [FNV17] (even without applying our generic transformation above), and thus, implies a witness encryption scheme for NP in the generic group model. While the hypothesis we rely on is unproven, there are no known barriers for extending the current hardness of approximation results for the minimal distance problem to the more challenging parameter regime needed for our construction [Kho20].

Our 1-element laconic argument relies on the NP-hardness of approximating the minimal distance of a linear code (GapMDP). For an approximation factor $\beta > 0$, a GapMDP_β instance $(\mathbf{A}, d)$ consists of a matrix $\mathbf{A}$ (over a finite field $\mathbb{F}$) and a distance $d \in \mathbb{N}$. The problem is to decide whether the minimum distance (under the Hamming metric) of the code generated by $\mathbf{A}$ is at most d or greater than $\beta \cdot d$. Equivalently, we can use the following dual formulation. An input instance is a pair $(\mathbf{H}, d)$, for a parity-check matrix $\mathbf{H} \in \mathbb{F}^{\ell \times k}$ (corresponding to the code generated by $\mathbf{A}$ in the previous formulation) and a distance bound d. The goal is to decide between the following two cases:

- YES instance: there exists $\mathbf{0} \neq \mathbf{v} \in \mathbb{F}_p^\ell$ with Hamming weight at most d such that $\mathbf{Hv} = \mathbf{0}$.
- NO instance: For all $\mathbf{0} \neq \mathbf{v} \in \mathbb{F}_p^\ell$ with Hamming weight $\leq \beta \cdot d$, we have $\mathbf{Hv} \neq \mathbf{0}$.

Note that GapMDP can be viewed as a homogeneous variant of GapMWSP. Based on the hypothesis that $\mathsf{GapMWSP}_\beta$ is NP-hard for some $\beta = \omega(\log n)$ and fields $\mathbb{F}_p$ of super-polynomial size, we construct a 1-element

[8] In a witness encryption scheme [GGSW13], the prover can encrypt a message to an NP statement x such that anyone with knowledge of the witness w is able to decrypt and recover the message. This is a very powerful notion of encryption whose only instantiations rely on indistinguishability obfuscation [GGH+13] (see Section 1.1 for recent developments), multilinear maps [GGSW13, GLW14, CVW18], or new and relatively unexplored algebraic structures [BIJ+20].

laconic argument for the GapMDP_β language with negligible soundness error in the generic group model. We use the same principles we used to construct the 1-query linear PCP for $\mathsf{GapMWSP}_\beta$. The construction operates over a group $\mathbb{G}$ of prime order p with generator g (which we will model as a generic group for the security analysis). The construction works as follows:

- **Query generation:** The verifier samples random vectors $\mathbf{c} \xleftarrow{R} \mathbb{F}_p^k$ $\mathbf{r} \xleftarrow{R} \mathbb{F}_p^k$ and a scalar $s \xleftarrow{R} \mathbb{F}_p$. It also samples a noise vector $\mathbf{e} \in \mathbb{F}_p^k$, where the entries of $\mathbf{e}$ are either 0 or uniform over $\mathbb{F}_p$. The density of $\mathbf{e}$ is chosen to balance the completeness and soundness requirements. The verifier computes $\mathbf{z}^\mathsf{T} = \mathbf{r}^\mathsf{T}\mathbf{H} + s\mathbf{c}^\mathsf{T} + \mathbf{e}^\mathsf{T} \in \mathbb{F}_p^k$. The query is the pair $(\mathbf{c}, g^{\mathbf{z}})$ where $g^{\mathbf{z}}$ denotes the vector of group elements $(g^{z_1}, \ldots, g^{z_k})$, and $\mathbf{z} = (z_1, \ldots, z_k)$.
- **Prover's response:** For a YES instance $(\mathbf{H}, d)$ to the GapMDP problem, the witness is a nonzero vector $\mathbf{v} \in \mathbb{F}_p^k$ such that $\mathbf{Hv} = \mathbf{0}$ and $\mathbf{v}$ has low Hamming weight. On input the query $\mathbf{c}$ and $g^{\mathbf{z}}$ and the witness $\mathbf{v} \in \mathbb{F}_p^k$, the prover computes $t = (\mathbf{c}^\mathsf{T}\mathbf{v})^{-1}$ and replies with the single group element $g^{t \cdot \mathbf{z}^\mathsf{T}\mathbf{v}}$.
- **Verification:** To verify the proof π, the verifier checks that $\pi = g^s$.

We now informally describe the completeness and soundness analysis:

- **Completeness:** For a YES instance, the (nonzero) witness $\mathbf{v}$ satisfies $\mathbf{Hv} = \mathbf{0}$ and moreover $\mathbf{v}$ has low Hamming weight. If the noise vector $\mathbf{e}$ is sufficiently sparse, then with high probability, $\mathbf{e}^\mathsf{T}\mathbf{v} = 0$. Thus,
$$\mathbf{z}^\mathsf{T}\mathbf{v} = \mathbf{r}^\mathsf{T}\mathbf{H}\mathbf{v} + s\mathbf{c}^\mathsf{T}\mathbf{v} + \mathbf{e}^\mathsf{T}\mathbf{v} = s\mathbf{c}^\mathsf{T}\mathbf{v}.$$
In this case, $g^{t \cdot \mathbf{z}^\mathsf{T}\mathbf{v}} = g^s$ since $t = (\mathbf{c}^\mathsf{T}\mathbf{v})^{-1}$. Note that since $\mathbf{c}$ is uniform (and independent of $\mathbf{v}$), and $\mathbf{v} \neq \mathbf{0}$, the scalar $\mathbf{c}^\mathsf{T}\mathbf{v}$ is nonzero with overwhelming probability, and thus invertible.
- **Soundness:** For the soundness analysis, we model the group as a generic group. Since the prover only has an encoding $g^{\mathbf{z}}$ of $\mathbf{z} \in \mathbb{F}_p^k$, in the generic group model, the only components that it can construct are of the form $g^{\mathbf{z}^\mathsf{T}\mathbf{w}+\delta}$ for some choice of $\mathbf{w} \in \mathbb{F}_p^k$ and $\delta \in \mathbb{F}_p$. Here, $\mathbf{w}$ and δ can depend on the parity-check matrix $\mathbf{H}$ and the vector $\mathbf{c}$, but can be considered to be independent of $\mathbf{r}$, $\mathbf{e}$, and s in the generic group model (we refer to Section 5.2 for the formal analysis). The prover succeeds if $\mathbf{z}^\mathsf{T}\mathbf{w} + \delta = \mathbf{r}^\mathsf{T}\mathbf{H}\mathbf{w} + s\mathbf{c}^\mathsf{T}\mathbf{w} + \mathbf{e}^\mathsf{T}\mathbf{w} = s$. We consider three possibilities:
  - If $\mathbf{w} = \mathbf{0}$, then $\mathbf{z}^\mathsf{T}\mathbf{w} + \delta = \delta$. Since s is uniform over $\mathbb{F}_p$ (and independent of δ), the verifier rejects with probability $1 - 1/p$.
  - If $\mathbf{w} \neq \mathbf{0}$, the Hamming weight of $\mathbf{w}$ is at most $\beta \cdot d$, and we have a NO instance, then $\mathbf{Hw} \neq \mathbf{0}$. In this case, over the randomness of $\mathbf{r}$ (which is uniform and independent of $\mathbf{H}$ and $\mathbf{w}$), the value of $\mathbf{r}^\mathsf{T}\mathbf{H}\mathbf{w} \neq 0$ is uniform over $\mathbb{F}_p$ and the verifier rejects with probability $1 - 1/p$.
  - Alternatively, if $\mathbf{w}$ has Hamming weight larger than $\beta \cdot d$, and $\mathbf{e}$ is sufficiently dense (and independent of $\mathbf{w}$), then with overwhelming probability, there is some component e_i such that $e_i w_i \neq 0$, and so $\mathbf{e}^\mathsf{T}\mathbf{w} \neq 0$. In this case, the value of $\mathbf{e}^\mathsf{T}\mathbf{w}$ is uniform over $\mathbb{F}_p$ and the verifier rejects with probability $1 - 1/p$.

  This means that for any choice of $\mathbf{w}, \delta$ that the prover chooses, the probability that $\mathbf{z}^\mathsf{T}\mathbf{w} + \delta = s$ is negligible (over the randomness of $\mathbf{r}$, $\mathbf{e}$, and s).

Observe that in the above analysis, we require $\mathbf{e}$ to be sufficiently sparse for completeness to hold with high probability and sufficiently dense for soundness to hold with overwhelming probability. For this reason, we require that the gap β be large enough so as to satisfy both constraints. In particular, taking $\beta = \omega(\log n)$ suffices for our analysis. We provide the full description of the scheme and its analysis in Section 5.2.

To obtain a 1-element laconic argument for NP (and correspondingly, a witness encryption scheme for NP), we need to assume that the above GapMDP_β problem is NP-hard for some choice of $\beta = \omega(\log n)$ and $\mathbb{F}_p$

is a finite field of super-polynomial size. More precisely, we require that there is a deterministic polynomial-time Karp-Levin reduction from NP to GapMDP_β (i.e., there exists an efficient algorithm that maps an NP statement to a GapMDP_β instance and a witness for the NP statement to a witness for the GapMDP_β instance). Existing hardness of approximation results show that over polynomial-size fields, the GapMDP problem is NP-hard for constant approximation factors $\beta = O(1)$, and NP-hard under a deterministic *quasi-polynomial* time reduction for "almost-polynomial" approximation factors $\beta = 2^{\log^{1-\varepsilon}(n)}$. Thus, proving our hypothesis (Hypothesis 5.12) requires strengthening existing hardness results in two directions: (1) arguing NP-hardness for some $\beta = \omega(\log n)$ under a polynomial-time reduction; and (2) extending the hardness result to super-polynomial prime order fields. As mentioned above, while our existing techniques do not seem sufficient, there are also no known barriers to showing the hardness of approximation results we require [Kho20]. If our hypothesis is true, then we obtain an *unconditional* construction of witness encryption for NP in the generic group model.

Witness encryption via GapMWSP? While GapMDP and GapMWSP are conceptually similar, it seems challenging to obtain a 1-element laconic argument (and correspondingly, a witness encryption scheme) directly from GapMWSP. In particular, if we embed the linear PCP query $\mathbf{q}^\mathsf{T} = \mathbf{r}^\mathsf{T}\mathbf{A} + \mathbf{e}^\mathsf{T}$ from the GapMWSP construction in the exponent (in a similar manner as above) and the prover is able to find an input $\mathbf{x}$ with low Hamming weight where $\mathbf{A}\mathbf{x} = \mathbf{0}$, then the prover learns $g^{\mathbf{e}^\mathsf{T}\mathbf{x}}$. Since both $\mathbf{e}$ and $\mathbf{x}$ have low Hamming weight, the inner product $\mathbf{e}^\mathsf{T}\mathbf{x}$ will be 0 with noticeable probability; this leaks information about the non-zero indices of $\mathbf{e}$. Observe that the GapMDP problem explicitly rules out the existence of nonzero vectors $\mathbf{x}$ with low Hamming weight where $\mathbf{A}\mathbf{x} = \mathbf{0}$. We provide more details in Remark 5.19.

However, we note that if we have an ideal obfuscation scheme [BGI+01] for membership testing for affine subsets of $\mathbb{F}^n$, then it is possible to obtain a 1-element laconic argument as well as a witness encryption scheme for NP *without* needing Hypothesis 5.12. The affine subset membership testing program we need to obfuscate closely resembles the types of programs supported by lockable obfuscation or compute-and-compare obfuscation [GKW17, WZ17]. Unfortunately, as we discuss in Remark 5.20, the existing constructions do not currently suffice to realize a witness encryption scheme. Nonetheless, the approach we develop in this work may provide another direction towards constructing witness encryption (without needing Hypothesis 5.12).

2 Preliminaries

For a positive integer $n \in \mathbb{N}$, we write $[n]$ to denote the set $\{1, \ldots, n\}$. We write $\mathbb{F}$ to denote a finite field. We will use bold lowercase letters (e.g., $\mathbf{v}, \mathbf{w}$) to denote vectors and bold uppercase letters (e.g., $\mathbf{A}, \mathbf{B}$) to denote matrices. For a vector $\mathbf{v} \in \mathbb{F}^n$, $\mathsf{wt}(\mathbf{v})$ denotes the Hamming weight of $\mathbf{v}$ (i.e., the number of nonzero entries in $\mathbf{v}$). For a matrix $\mathbf{A} \in \mathbb{F}^{n \times m}$, we write $\mathsf{dist}(\mathbf{A})$ to denote the minimum distance of the code generated by $\mathbf{A}$. (i.e., the minimum Hamming weight of a nonzero codeword generated by $\mathbf{A}$).

We write λ to denote a security parameter. We say that a function f is negligible in λ, denoted $\mathsf{negl}(\lambda)$ if $f(\lambda) = o(1/\lambda^c)$ for all $c \in \mathbb{N}$. We say an event happens with negligible probability if the probability of the event happening is negligible, and that it happens with overwhelming probability if its complement occurs with overwhelming probability. We say an algorithm is efficient if it runs in probabilistic polynomial time in the length of its input. We write $\mathsf{poly}(\lambda)$ to denote a function that is bounded by a fixed polynomial in λ and $\mathsf{polylog}(\lambda)$ to denote a function that is bounded by $\mathsf{poly}(\log \lambda)$. We say that two families of distributions $\mathcal{D}_1 = \{\mathcal{D}_{1,\lambda}\}_{\lambda \in \mathbb{N}}$ and $\mathcal{D}_2 = \{\mathcal{D}_{2,\lambda}\}_{\lambda \in \mathbb{N}}$ are computationally indistinguishable (denoted $\mathcal{D}_1 \overset{c}{\approx} \mathcal{D}_2$) if no efficient adversary can distinguish samples from $\mathcal{D}_1$ and $\mathcal{D}_2$ except with negligible probability. We say that $\mathcal{D}_1$ and $\mathcal{D}_2$ are statistically indistinguishable (denoted $\mathcal{D}_1 \overset{s}{\approx} \mathcal{D}_2$) if the statistical distance between $\mathcal{D}_1$ and $\mathcal{D}_2$ is negligible. We will also use the Schwartz-Zippel lemma and Hoeffding's inequality in our analysis:

Lemma 2.1 (Schwartz-Zippel [Sch80, Zip79])**.** *Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be a multivariate polynomial of total degree d over $\mathbb{F}$, not identically zero. Then for any set $S \subseteq \mathbb{F}$,*

$$\Pr[f(\alpha_1, \ldots, \alpha_n) = 0 \mid \alpha_1, \ldots, \alpha_n \xleftarrow{R} S] \leq d/|S|.$$

Fact 2.2 (Hoeffding's Inequality [Hoe63])**.** Let $X_1, \ldots, X_n$ be independent random variables where $X_i \in [-B, B]$ for some bound $B > 0$. Let $S_n = \sum_{i \in [n]} X_i$. Then,

$$\Pr[|S_n - \mathbb{E}[S_n]| \geq t] \leq 2 \exp\left(-\frac{2t^2}{nB^2}\right)$$

NP problems and reductions. We say that there is a Karp reduction from a problem $\Pi \subseteq \{0,1\}^*$ to a problem $\Pi' \subseteq \{0,1\}^*$ if there exists an efficiently-computable mapping f such that $x \in \Pi$ if and only if $f(x) \in \Pi'$. When $\Pi, \Pi' \in \mathsf{NP}$ with associated NP relations $\mathcal{R}, \mathcal{R}'$, respectively, we say that there is a Karp-Levin reduction from $\mathcal{R}$ to $\mathcal{R}'$ (or from Π to Π') if there additionally exists an efficiently-computable mapping g such that $\mathcal{R}(x, w) = 1$ if and only if $\mathcal{R}'(f(x), g(x, w)) = 1$.[9] We note that natural NP-complete relations are typically NP-complete also with respect to the stronger notion of Karp-Levin reduction.

In this work we will also consider *promise problems* Π where $\Pi = (\Pi_{\text{YES}}, \Pi_{\text{NO}})$ is a pair of disjoint sets $\Pi_{\text{YES}}, \Pi_{\text{NO}} \subseteq \{0,1\}^*$. The union $\Pi_{\text{YES}} \cup \Pi_{\text{NO}}$ is called the "promise." Elements of Π_{YES} are YES instances and elements of Π_{NO} are NO instances. Every NP promise problem $\Pi = (\Pi_{\text{YES}}, \Pi_{\text{NO}})$ is additionally characterized by an efficiently-computable relation $\mathcal{R}$ with the following properties:

- For all $x \in \Pi_{\text{YES}}$, there exists a witness $w \in \{0,1\}^{\mathsf{poly}(|x|)}$ such that $\mathcal{R}(x, w) = 1$;
- For all $x \in \Pi_{\text{NO}}$ and all $w \in \{0,1\}^*$, $\mathcal{R}(x, w) = 0$.

Note in particular that there are no guarantees on the behavior of $\mathcal{R}(x, \cdot)$ when $x \notin \Pi_{\text{YES}} \cup \Pi_{\text{NO}}$. We say that there is a Karp reduction from an NP problem Π to an NP promise problem $\Pi' = (\Pi'_{\text{YES}}, \Pi'_{\text{NO}})$ if there exists an efficiently-computable mapping f such that $x \in \Pi \implies f(x) \in \mathcal{L}'_{\text{YES}}$ and $x \notin \Pi \implies f(x) \in \Pi'_{\text{NO}}$. In particular, the function f always outputs either a YES instance or a NO instance for the promise problem. Similarly, if $\mathcal{R}$ is the NP relation associated with Π and $\mathcal{R}'$ is the NP relation associated with the NP promise problem Π', we say that there is a Karp-Levin reduction from $\mathcal{R}$ to $\mathcal{R}'$ if in addition to the function f, there also exists an efficiently-computable function g such that for all $x \in \Pi$, $\mathcal{R}(x, w) = 1$ if and only if $\mathcal{R}'(f(x), g(x, w)) = 1$. Much like the case with reductions between NP-complete problems, many natural hardness of approximation results for NP optimization problems also hold with respect to the stronger notion of Karp-Levin reductions.

Prime order groups. All of our cryptographic constructions in this work are defined over a *pairing-free* group $\mathbb{G}$ of prime order p. We define the notion of a prime-order group generator below.

Definition 2.3 (Prime-Order Group Generator)**.** A prime-order group generator algorithm $\mathsf{GroupGen}$ is an efficient algorithm that on input the security parameter 1^λ outputs a description $\mathcal{G} = (\mathbb{G}, p, g)$ of a prime-order group $\mathbb{G}$ with order p and generator g. Throughout this work, we will assume that $p = 2^{\Theta(\lambda)}$.

In our analysis, we will sometimes model the group $\mathbb{G}$ as a "generic group" [Nec94, Sho97]. We provide the definition of the generic group model in Appendix C. Additionally, we recall the definition of a succinct non-interactive argument, laconic arguments, as well as the [BCI+13] compiler in Appendix A.

Arithmetic circuit satisfiability. A central part of this work is constructing succinct argument systems for the language of Boolean circuit satisfiability. When describing some of our constructions however, it will oftentimes be more natural to consider the more *general* language of *arithmetic circuit satisfiability* which we recall formally below. Throughout this paper, an arithmetic circuit $C \colon \mathbb{F}^n \times \mathbb{F}^h \to \mathbb{F}^\ell$ over a finite field $\mathbb{F}$ consists of a collection of addition gates with unbounded fan-in and multiplication gates with fan-in 2. Both types of gates can have unbounded fan-out. As noted in [BCI+13], Boolean circuit satisfiability can be reduced to arithmetic circuit satisfiability over any finite field $\mathbb{F}$ with constant overhead.

[9] Typically, a Levin reduction also requires an efficiently-computable mapping h such that $\mathcal{R}'(f(x), z) = 1$ if and only if $\mathcal{R}(x, h(x, z)) = 1$, but our constructions in this work will not require this additional property.

Definition 2.4 (Arithmetic Circuit Satisfiability)**.** Let $\mathbb{F}$ be a finite field. For an arithmetic circuit $C\colon \mathbb{F}^n \times \mathbb{F}^h \to \mathbb{F}^t$ over $\mathbb{F}$, the *arithmetic circuit satisfiability* problem is defined by the relation $\mathcal{R}_C = \{(\mathbf{x}, \mathbf{w}) \in \mathbb{F}^n \times \mathbb{F}^h : C(\mathbf{x}, \mathbf{w}) = 0^t\}$. We write $\mathcal{L}_C$ to denote the corresponding language. For a family of arithmetic circuits $\mathcal{C} = \{C_\ell\colon \mathbb{F}^{n(\ell)} \times \mathbb{F}^{h(\ell)} \to \mathbb{F}^{t(\ell)}\}_{\ell \in \mathbb{N}}$, we write $\mathcal{R}_\mathcal{C}$ and $\mathcal{L}_\mathcal{C}$ to denote the infinite relation $\mathcal{R}_\mathcal{C} = \bigcup_{\ell \in \mathbb{N}} \mathcal{R}_{C_\ell}$ and infinite language $\mathcal{L}_\mathcal{C} = \bigcup_{\ell \in \mathbb{N}} \mathcal{L}_{C_\ell}$. The special case of *Boolean circuit satisfiability* is the problem of arithmetic circuit satisfiability over the binary field $\mathbb{F} = \mathbb{F}_2$ (in this case, the output of C can be taken to be a single bit (i.e., $\ell = 1$) without loss of generality).

2.1 Linear PCPs

We begin by recalling the definition of linear PCPs (LPCP) from [BCI+13]. Our definition combines features from a "fully linear PCP" introduced in [BBC+19] with the traditional notion of a linear PCP. First, recall that in a fully linear PCP, the verifier does not have direct access to the statement $\mathbf{x} \in \mathbb{F}^n$ and instead is given linear query access to the vector $[\boldsymbol{\pi}, \mathbf{x}]$ that includes the proof $\boldsymbol{\pi}$ together with the statement $\mathbf{x}$. To simplify the definition (and still capture existing constructions of linear PCPs), in a k-query linear PCP, we allow the verifier to make a single "free" linear query to the statement $\mathbf{x}$ and up to k linear queries to the proof vector $\boldsymbol{\pi}$. We give our definition below:

Definition 2.5 (Linear PCP [BCI+13, BBC+19, adapted])**.** Let $\mathcal{R}\colon \mathbb{F}^n \times \mathbb{F}^h \to \{0,1\}$ be a binary relation[10] (with associated language $\mathcal{L}$) over a finite field $\mathbb{F}$. A k-query linear PCP for $\mathcal{R}$ with query length ℓ and soundness error ε is a tuple of algorithms $\Pi_{\mathsf{LPCP}} = (\mathcal{Q}_{\mathsf{LPCP}}, \mathcal{P}_{\mathsf{LPCP}}, \mathcal{D}_{\mathsf{LPCP}})$ with the following properties:

- The verifier's query algorithm $\mathcal{Q}_{\mathsf{LPCP}}$ outputs a query $\mathbf{q}_{\mathsf{inp}} \in \mathbb{F}^n$, a query matrix $\mathbf{Q} \in \mathbb{F}^{\ell \times k}$, and a verification state st. We can also consider an *input-dependent* linear PCP where the query algorithm also takes as input a statement $\mathbf{x} \in \mathbb{F}^n$.
- The prover algorithm $\mathcal{P}_{\mathsf{LPCP}}$ takes a statement $\mathbf{x} \in \mathbb{F}^n$ and a witness $\mathbf{w} \in \mathbb{F}^h$ as input and outputs a proof $\boldsymbol{\pi} \in \mathbb{F}^\ell$.
- The verifier's decision algorithm $\mathcal{D}_{\mathsf{LPCP}}$ takes as input the verification state st, an input-dependent response $a_{\mathsf{inp}} \in \mathbb{F}$, and a vector of responses $\mathbf{a} \in \mathbb{F}^k$, and outputs a bit $b \in \{0,1\}$.

In addition, Π_{LPCP} should satisfy the following properties:

- **Completeness:** For all $\mathbf{x} \in \mathbb{F}^n$ and $\mathbf{w} \in \mathbb{F}^h$ where $\mathcal{R}(\mathbf{x}, \mathbf{w}) = 1$,

$$\Pr[\mathcal{D}_{\mathsf{LPCP}}(\mathsf{st}, \mathbf{q}_{\mathsf{inp}}^\mathsf{T}\mathbf{x}, \mathbf{Q}^\mathsf{T}\boldsymbol{\pi}) = 1 \mid (\mathsf{st}, \mathbf{q}_{\mathsf{inp}}, \mathbf{Q}) \leftarrow \mathcal{Q}_{\mathsf{LPCP}}, \boldsymbol{\pi} \leftarrow \mathcal{P}_{\mathsf{LPCP}}(\mathbf{x}, \mathbf{w})] = 1$$

- **Soundness:** For every $\mathbf{x} \notin \mathcal{L}$ and every $\boldsymbol{\pi}^* \in \mathbb{F}^\ell$, $\boldsymbol{\delta}^* \in \mathbb{F}^k$,

$$\Pr[\mathcal{D}_{\mathsf{LPCP}}(\mathsf{st}, \mathbf{q}_{\mathsf{inp}}^\mathsf{T}\mathbf{x}, \mathbf{Q}^\mathsf{T}\boldsymbol{\pi}^* + \boldsymbol{\delta}^*) = 1 \mid (\mathsf{st}, \mathbf{q}_{\mathsf{inp}}, \mathbf{Q}) \leftarrow \mathcal{Q}_{\mathsf{LPCP}}] \leq \varepsilon.$$

  We refer to ε as the *soundness error*.

- **δ-Honest-verifier zero-knowledge (δ-HVZK):** There exists an efficient simulator $\mathcal{S}_{\mathsf{LPCP}}$ such that for all $\mathbf{x} \in \mathcal{L}$, the following distributions are δ-close (i.e., their statistical distance is at most δ):

$$\{\mathcal{S}_{\mathsf{LPCP}}(\mathbf{x})\} \text{ and } \left\{(\mathsf{st}, \mathbf{q}_{\mathsf{inp}}, \mathbf{Q}, \mathbf{q}_{\mathsf{inp}}^\mathsf{T}\mathbf{x}, \mathbf{Q}^\mathsf{T}\boldsymbol{\pi}) \;\middle|\; \begin{array}{c}(\mathsf{st}, \mathbf{q}_{\mathsf{inp}}, \mathbf{Q}) \leftarrow \mathcal{Q}_{\mathsf{LPCP}}; \\ \boldsymbol{\pi} \leftarrow \mathcal{P}_{\mathsf{LPCP}}(\mathbf{x}, \mathbf{w})\end{array}\right\}.$$

[10] We can also define integer linear PCPs for an (infinite) family of relations $\mathcal{R} = \bigcup_{\kappa \in \mathbb{N}} \mathcal{R}_\kappa$. In this case, the inputs to the query-generation and proving algorithms would additionally take the relation index 1^κ as input, and the parameters $n, h, k, \ell, B, \varepsilon$ can all be functions of κ.

If these two distributions are identically distributed, we say that LPCP satisfies *perfect honest-verifier zero-knowledge.*[11]

Definition 2.6 (Strong Soundness [BCI+13])**.** A k-query linear PCP $\Pi_{\mathsf{LPCP}} = (\mathcal{Q}_{\mathsf{LPCP}}, \mathcal{P}_{\mathsf{LPCP}}, \mathcal{D}_{\mathsf{LPCP}})$ satisfies ε-*strong soundness* if it has soundness error at most ε, and moreover, for every input $\mathbf{x} \in \mathbb{F}^n$ and every affine function $\boldsymbol{\pi}^* \in \mathbb{F}^\ell$, $\boldsymbol{\delta}^* \in \mathbb{F}^k$, either

$$\Pr[\mathcal{D}_{\mathsf{LPCP}}(\mathsf{st}, \mathbf{q}_{\mathsf{inp}}^\mathsf{T}\mathbf{x}, \mathbf{Q}^\mathsf{T}\boldsymbol{\pi}^* + \boldsymbol{\delta}^*) = 1 \mid (\mathsf{st}, \mathbf{q}_{\mathsf{inp}}, \mathbf{Q}) \leftarrow \mathcal{Q}_{\mathsf{LPCP}}] \leq \varepsilon,$$

or

$$\Pr[\mathcal{D}_{\mathsf{LPCP}}(\mathsf{st}, \mathbf{q}_{\mathsf{inp}}^\mathsf{T}\mathbf{x}, \mathbf{Q}^\mathsf{T}\boldsymbol{\pi}^* + \boldsymbol{\delta}^*) = 1 \mid (\mathsf{st}, \mathbf{q}_{\mathsf{inp}}, \mathbf{Q}) \leftarrow \mathcal{Q}_{\mathsf{LPCP}}] = 1.$$

Definition 2.7 (Knowledge Soundness)**.** A k-query linear PCP $\Pi_{\mathsf{LPCP}} = (\mathcal{Q}_{\mathsf{LPCP}}, \mathcal{P}_{\mathsf{LPCP}}, \mathcal{D}_{\mathsf{LPCP}})$ for a relation $\mathcal{R}$ over a finite field $\mathbb{F}$ satisfies ε-knowledge soundness if there exists a knowledge extractor $\mathcal{E}$ such that for every affine function $\boldsymbol{\pi}^* \in \mathbb{F}^\ell$, $\boldsymbol{\delta}^* \in \mathbb{F}^k$, if

$$\Pr[\mathcal{D}_{\mathsf{LPCP}}(\mathsf{st}, \mathbf{q}_{\mathsf{inp}}^\mathsf{T}\mathbf{x}, \mathbf{Q}^\mathsf{T}\boldsymbol{\pi}^* + \boldsymbol{\delta}^*) = 1 \mid (\mathsf{st}, \mathbf{q}_{\mathsf{inp}}, \mathbf{Q}) \leftarrow \mathcal{Q}_{\mathsf{LPCP}}] \geq \varepsilon,$$

then $\mathcal{E}^{\langle \boldsymbol{\pi}^*, \cdot \rangle + \boldsymbol{\delta}^*}(\mathbf{x})$ outputs $\mathbf{w}$ such that $(\mathbf{x}, \mathbf{w}) \in \mathcal{R}$.

Definition 2.8 (Algebraic Degree of Verifier)**.** We say that the algebraic degree of a linear PCP $\Pi_{\mathsf{LPCP}} = (\mathcal{Q}_{\mathsf{LPCP}}, \mathcal{P}_{\mathsf{LPCP}}, \mathcal{D}_{\mathsf{LPCP}})$ is d if the verifier's decision algorithm $\mathcal{D}_{\mathsf{LPCP}}$ can be computed by a multivariate polynomial (over $\mathbb{F}$) of total degree d in its inputs.

3 1-Query Linear PCPs via Packing

In this section, we begin by introducing the notion of a bounded linear PCP over the finite field $\mathbb{F}_p$. Throughout this section, we will view elements $x \in \mathbb{F}_p$ as both field elements over $\mathbb{F}_p$ as well as integers in the interval $[-p/2, p/2]$. We first show in Construction 3.4 how to pack k-query bounded linear PCPs into a 1-query linear PCP. In Appendix B, we describe how to construct a 2-query linear PCP based on the Hadamard linear PCP [ALM+98, IKO07]. In conjunction with our query-packing transformation, we obtain 1-query linear PCPs for NP. Then, by invoking the compiler from [BCI+13] (see Appendix A.1) with the ElGamal encryption scheme, we obtain a SNARG where the proof consists of a single ElGamal ciphertext (i.e., two group elements). Then, in Sections 3.2 and 3.3, we show how to optimize the *concrete efficiency* of our ElGamal-based SNARG (by leveraging structural properties of our 1-query linear PCP).

Definition 3.1 (Bounded Linear PCP)**.** A k-query linear PCP $\Pi_{\mathsf{LPCP}} = (\mathcal{Q}_{\mathsf{LPCP}}, \mathcal{P}_{\mathsf{LPCP}}, \mathcal{D}_{\mathsf{LPCP}})$ for a relation $\mathcal{R} \colon \mathbb{F}_p^n \times \mathbb{F}_p^h \to \{0,1\}$ over a finite field $\mathbb{F}_p$ is *bounded* with respect to bound functions $b_1, \ldots, b_k \colon \mathbb{N} \to \mathbb{N}$ if $\mathcal{Q}_{\mathsf{LPCP}}$ and $\mathcal{P}_{\mathsf{LPCP}}$ take as input an additional bound parameter $\tau \in \mathbb{N}$ and for any $\mathbf{x}, \mathbf{w}$ where $\mathcal{R}(\mathbf{x}, \mathbf{w}) = 1$, we have for all $i \in [k]$,

$$\Pr[\mathbf{q}_i^\mathsf{T}\boldsymbol{\pi} \in [-b_i(\tau), b_i(\tau)] \mid (\mathsf{st}, \mathbf{q}_{\mathsf{inp}}, \mathbf{Q}) \leftarrow \mathcal{Q}_{\mathsf{LPCP}}(\tau), \boldsymbol{\pi} \leftarrow \mathcal{P}_{\mathsf{LPCP}}(\tau, \mathbf{x}, \mathbf{w})] = 1, \tag{3.1}$$

where $\mathbf{q}_i$ denotes the i^{th} column on $\mathbf{Q}$ and the inner product is computed over the *integers*. We say that Π_{LPCP} is bounded with respect to bound functions $b_1, \ldots, b_k$ with probability ε if Eq. (3.1) holds with probability ε. Moreover, when defining δ-HVZK for bounded linear PCPs, we additionally provide the bound parameter τ as input to $\mathcal{Q}_{\mathsf{LPCP}}$ and $\mathcal{P}_{\mathsf{LPCP}}$ in the real distribution and the simulator $\mathcal{S}_{\mathsf{LPCP}}$ in the simulated distribution (in addition to the input x). In this case, we also allow the bound function to depend on both the bound parameter τ as well as the zero-knowledge parameter δ.

[11] We can consider a stronger notion of zero-knowledge where the simulator does not have access to the statement $\mathbf{x}$. This is the setting considered in fully linear PCPs [BBC+19] and has applications to constructing proofs on committed values or secret-shared values. This stronger notion can also be relevant in our setting where a verifier is checking proofs from multiple provers (who may each hold a secret share of a distributed database), and the goal is to minimize proof size or verifier complexity.

Remark 3.2 (Responses that Exceed the Bound)**.** A bounded LPCP has the guarantee that the responses computed using an *honestly-generated* proof will be bounded. However, this property does not extend to responses computed with respect to a maliciously-generated proof. In a bounded LPCP, the verifier has the option to explicitly enforce the bound and *only* accept proofs where the responses fall within the prescribed bound, or it can choose not to enforce the bound. This choice does not affect completeness, soundness or honest-verifier zero-knowledge, but it does affect *strong soundness*. Namely, for *true* statements, there can be responses that the verifier accepts and yet, the values fall outside the bound (these responses correspond to proof strategies that are never output by the honest algorithm). Moreover, whether the responses computed by such strategies lie within the bound or not can be correlated with the verifier's queries. As such, a verifier that does not enforce the bound might satisfy strong soundness whereas a verifier that does enforce the bound may fail to satisfy strong soundness. In our linear PCP packing construction (Construction 3.4), strong soundness is only preserved if we assume that the underlying bounded linear PCPs satisfy strong soundness for verifiers that enforce the bound. For this reason, we define this requirement explicitly in Definition 3.3.

Definition 3.3 (Strong Soundness for Bounded LPCPs)**.** A k-query bounded linear PCP $\Pi_{\mathsf{LPCP}} = (\mathcal{Q}_{\mathsf{LPCP}}, \mathcal{P}_{\mathsf{LPCP}}, \mathcal{D}_{\mathsf{LPCP}})$ with bound functions $b_1, \ldots, b_k$ satisfies ε-strong soundness if it satisfies the usual notion of ε-strong soundness (Definition 2.6) and $\mathcal{D}_{\mathsf{LPCP}}(\mathsf{st}, a_{\mathsf{inp}}, \mathbf{a})$ outputs 1 only if $a_i \in [-b_i(\tau), b_i(\tau)]$ for $i \in [k]$ when $(\mathsf{st}, \mathbf{q}_{\mathsf{inp}}, \mathbf{Q}) \leftarrow \mathcal{Q}_{\mathsf{LPCP}}(\tau)$.

Construction 3.4 (Bounded Linear PCP Packing)**.** Let $\Pi'_{\mathsf{LPCP}} = (\mathcal{Q}'_{\mathsf{LPCP}}, \mathcal{P}'_{\mathsf{LPCP}}, \mathcal{D}'_{\mathsf{LPCP}})$ be a k-query bounded LPCP for a binary relation $\mathcal{R}\colon \mathbb{F}_p^n \times \mathbb{F}_p^h \to \{0,1\}$ over $\mathbb{F}_p$ with bound functions $b'_1, \ldots, b'_k\colon \mathbb{N} \to \mathbb{N}$ and soundness error ε'. We will assume that $\mathcal{D}'_{\mathsf{LPCP}}$ strictly enforces the bound on the responses (namely, given a set of responses $a_1, \ldots, a_k$, $\mathcal{D}'_{\mathsf{LPCP}}$ accepts only if $a_i \in [-b'_i(\tau), b'_i(\tau)]$ for each $i \in [k]$; see Remark 3.2 and Definition 3.3 for more discussion). We construct a 1-query bounded LPCP $\Pi_{\mathsf{LPCP}} = (\mathcal{Q}_{\mathsf{LPCP}}, \mathcal{P}_{\mathsf{LPCP}}, \mathcal{D}_{\mathsf{LPCP}})$ as follows:

- $\mathcal{Q}_{\mathsf{LPCP}}(\tau)$: On input the bound parameter τ, $\mathcal{Q}_{\mathsf{LPCP}}$ proceeds as follows:

  1. Run $(\mathsf{st}', \mathbf{q}'_{\mathsf{inp}}, \mathbf{Q}') \leftarrow \mathcal{Q}'_{\mathsf{LPCP}}(\tau)$. Set $\mathbf{q}_{\mathsf{inp}} = \mathbf{q}'_{\mathsf{inp}}$.
  2. Define $B_{\mathsf{min}} = \min_{i\in[k]} b'_i(\tau)$, $B_{\mathsf{max}} = \max_{i\in[k]} b'_i(\tau)$, and $B_{\mathsf{mul}} = \prod_{i\in[k]} b'_i(\tau)$. Set $r_1 = 1$ and sample $r_2, \ldots, r_k \xleftarrow{\mathrm{R}} \left[4B_{\mathsf{max}} + 1, B_{\mathsf{mul}} \cdot 2^{k+2}/\varepsilon'\right]$. Without loss of generality, we will assume that $B_{\mathsf{min}} = b'_k(\tau)$.
  3. Compute $\mathsf{st} \leftarrow (\mathsf{st}', b'_1(\tau), \ldots, b'_k(\tau), r_1, \ldots, r_k)$, and compute the query vector $\mathbf{q} \in \mathbb{F}_p^\ell$ as

  $$\mathbf{q} = \sum_{i\in[k]} \mathbf{q}'_i \left(\prod_{j\in[i]} r_j \right) \in \mathbb{F}_p^\ell,$$

  where $\mathbf{q}'_1, \ldots, \mathbf{q}'_k \in \mathbb{Z}^\ell$ denote the k columns of $\mathbf{Q}'$.

  Output $(\mathsf{st}, \mathbf{q}_{\mathsf{inp}}, \mathbf{q})$.

- $\mathcal{P}_{\mathsf{LPCP}}(\tau, \mathbf{x}, \mathbf{w})$: On input a statement $\mathbf{x} \in \mathbb{F}_p^n$ and a witness $\mathbf{w} \in \mathbb{F}_p^h$, output the proof $\boldsymbol{\pi} \leftarrow \mathcal{P}_{\mathsf{LPCP}}(\tau, \mathbf{x}, \mathbf{w})$.

- $\mathcal{D}_{\mathsf{LPCP}}(\mathsf{st}, a_{\mathsf{inp}}, a)$: On input the state $\mathsf{st} = (\mathsf{st}', b'_1, \ldots, b'_k, r_1, \ldots, r_k)$, an input-dependent response $a_{\mathsf{inp}} \in \mathbb{F}_p$, and a response $a \in \mathbb{F}_p$, compute $a'_1, \ldots, a'_k \in \mathbb{F}_p$ so that $a = \sum_{i\in[k]} a'_i \prod_{j\in[i]} r_j$ and each a'_i satisfies $a'_i \in [-b'_i, b'_i]$. This can be done as follows:

  – For each $i = k, k-1, \ldots, 1$, compute $a'_i \leftarrow \lfloor a / \prod_{j\in[i]} r_j \rfloor$ and update $a \leftarrow a - a'_i \cdot \prod_{j\in[i]} r_j$, where all of these operations happen *over the integers* (namely, the algorithm interprets the values a and $a'_1, \ldots, a'_k$ as integers in the interval $[-p/2, p/2]$).

If the above procedure does not produce $a'_1, \ldots, a'_k \in \mathbb{F}_p$ satisfying the above requirements, output 0. Otherwise, output $\mathcal{D}'_{\mathsf{LPCP}}(\mathsf{st}', a_{\mathsf{inp}}, (a'_1, \ldots, a'_k))$.

Theorem 3.5 (Bounded)**.** *If Π'_{LPCP} is bounded with respect to bound functions $b'_1, \ldots, b'_k$ and $B_{\mathsf{mul}} \cdot 2^{k+2}/\varepsilon' \leq p/2$, then Π_{LPCP} in Construction 3.4 is bounded with respect to the bound*

$$b(\tau) = 2 \cdot B_{\mathsf{min}} \cdot (B_{\mathsf{mul}} \cdot 2^{k+2}/\varepsilon)^{k-1}.$$

Proof. Take any $(\mathbf{x}, \mathbf{w}) \in \mathcal{R}$, and let $(\mathsf{st}', \mathbf{q}'_{\mathsf{inp}}, \mathbf{Q}') \leftarrow \mathcal{Q}'_{\mathsf{LPCP}}(\tau)$, $\boldsymbol{\pi} \leftarrow \mathcal{P}_{\mathsf{LPCP}}(\tau, \mathbf{x}, \mathbf{w})$. By construction, $\mathbf{q} = \sum_{i \in [k]} \mathbf{q}'_i \left(\prod_{j \in [i]} r_j \right)$ where $\mathbf{q}'_1, \ldots, \mathbf{q}'_k$ are the columns of $\mathbf{Q}'$. Then,

$$\mathbf{q}^{\mathsf{T}} \boldsymbol{\pi} = \sum_{i \in [k]} \left[(\mathbf{q}'_i)^{\mathsf{T}} \boldsymbol{\pi} \prod_{j \in [i]} r_j \right] = \sum_{i \in [k]} \left[a'_i \prod_{j \in [i]} r_j \right] \in \mathbb{F}_p. \tag{3.2}$$

Since Π'_{LPCP} is bounded, $a'_i = (\mathbf{q}'_i)^{\mathsf{T}} \boldsymbol{\pi} \in [-b'_i, b'_i)]$ for all $i \in [k]$. Consider Eq. (3.2) over the *integers*, where we view the values $a'_1, \ldots, a'_k$ and $r_1, \ldots, r_k$ as integers in the range $[-p/2, p/2]$. The bound on a'_i means that each $|a'_i| \leq b'_i$ for all $i \in [k]$. Next, we note that by assumption, $r_1, \ldots, r_k \leq B_{\mathsf{mul}} \cdot 2^{k+2}/\varepsilon' < p/2$. Thus, over the integers, we now have

$$|\mathbf{q}^{\mathsf{T}} \boldsymbol{\pi}| \leq \sum_{i \in [k]} \left[|a'_i| \prod_{j \in [i]} r_j \right] \leq \sum_{i \in [k]} \left[b'_i \prod_{j \in [i]} r_j \right] \in \mathbb{Z}. \tag{3.3}$$

We now show that over the integers, for all $i \in [k]$,

$$\sum_{t \in [i]} b'_t \prod_{j \in [t]} r_j < \frac{1}{2} \prod_{j \in [i+1]} r_j. \tag{3.4}$$

We proceed inductively, Certainly this holds when $i = 1$ since $r_1 = 1$ and $r_2 \geq 4B_{\mathsf{max}} + 1 > 4b'_1$. Then,

$$\begin{aligned} \sum_{t \in [i+1]} b'_t \prod_{j \in [t]} r_j = \left(\sum_{t \in [i]} b'_t \prod_{j \in [t]} r_j \right) + \left(b'_{i+1} \prod_{j \in [i+1]} r_j \right) &< \left(\prod_{j \in [i+1]} r_j \right) + \left(b'_{i+1} \prod_{j \in [i+1]} r_j \right) \\ &= (b'_{i+1} + 1) \prod_{j \in [i+1]} r_j \leq \frac{1}{2} \prod_{j \in [i+2]} r_j. \end{aligned}$$

since $r_{i+2} \geq 4B_{\mathsf{max}} + 1 \geq 4(b'_{i+1} + 1)$. Applying this to Eq. (3.3), we have that

$$\begin{aligned} |\mathbf{q}^{\mathsf{T}} \boldsymbol{\pi}| \leq \sum_{i \in [k]} \left[b'_i \prod_{j \in [i]} r_j \right] &\leq \sum_{i \in [k-1]} \left[b'_i \prod_{j \in [i]} r_j \right] + b'_k \prod_{j \in [k]} r_j \leq \frac{1}{2} \prod_{j \in [k]} r_j + b'_k \prod_{j \in [k]} r_j \\ &\leq 2B_{\mathsf{min}} \prod_{j \in [k]} r_j \leq 2B_{\mathsf{min}} \left(B_{\mathsf{mul}} \cdot 2^{k+2}/\varepsilon' \right)^{k-1}, \end{aligned}$$

since $r_1 = 1$, $B_{\mathsf{min}} = b'_k$, and $r_2, \ldots, r_k \leq B_{\mathsf{mul}} \cdot 2^{k+2}/\varepsilon'$. Since this relation holds over the integers, the claim holds. □

Theorem 3.6 (Completeness)**.** *If Π'_{LPCP} is complete and bounded with respect to functions $b'_1, \ldots, b'_k$ and*

$$2B_{\mathsf{min}} \cdot \left(B_{\mathsf{mul}} 2^{k+2}/\varepsilon \right)^{k-1} \leq \frac{|\mathbb{F}_p|}{2} = \frac{p}{2},$$

then Π_{LPCP} from Construction 3.4 is complete.

Proof. Take any $(\mathbf{x}, \mathbf{w}) \in \mathcal{R}$, and let $(\mathsf{st}', \mathbf{q}'_{\mathsf{inp}}, \mathbf{q}') \leftarrow \mathcal{Q}'_{\mathsf{LPCP}}(\tau)$, $\boldsymbol{\pi} \leftarrow \mathcal{P}_{\mathsf{LPCP}}(\tau, \mathbf{x}, \mathbf{w})$. By construction, $\mathbf{q} = \sum_{i\in[k]} \mathbf{q}'_i \left(\prod_{j\in[i]} r_j\right)$ where $\mathbf{q}'_1, \ldots, \mathbf{q}'_k$ are the columns of $\mathbf{Q}'$ output by $\mathcal{Q}'_{\mathsf{LPCP}}$. By construction,

$$\mathbf{q}^{\mathsf{T}}\boldsymbol{\pi} = \sum_{i\in[k]} \left[(\mathbf{q}'_i)^{\mathsf{T}}\boldsymbol{\pi} \prod_{j\in[i]} r_j\right] = \sum_{i\in[k]} \left[a'_i \prod_{j\in[i]} r_j\right].$$

It suffices to argue that the decision algorithm correctly recover the values $a'_i = (\mathbf{q}'_i)^{\mathsf{T}}\boldsymbol{\pi}$ from a. The claim then follows from completeness of Π'_{LPCP}. As in the proof of Theorem 3.5, we view the values a, $a'_1, \ldots, a'_k$, and $r_1, \ldots, r_k$ as *integers* in the range $[-p/2, p/2]$. Then, it suffices to show for $i \in [k]$,

$$\prod_{j\in[i]} r_j > 2 \cdot \left| \sum_{t\in[i-1]} a'_t \prod_{j\in[t]} r_j \right| = 2 \cdot \sum_{t\in[i-1]} |a'_t| \prod_{j\in[t]} r_j,$$

since $\prod_{i\in[k]} r_i < p/2$. Since Π'_{LPCP} is bounded, $|a'_t| \le b'_t$ for all $t \in [k]$. This means that

$$\sum_{t\in[i-1]} |a'_t| \prod_{j\in[t]} r_j \le \sum_{t\in[i-1]} b'_t \prod_{j\in[t]} r_j < \frac{1}{2} \prod_{j\in[i]} r_j,$$

by Eq. (3.4), which was shown to hold for these parameters in the proof of Theorem 3.5. Thus, the verification algorithm correctly recovers the values of $a'_i = (\mathbf{q}'_i)^{\mathsf{T}}\boldsymbol{\pi}$ over the integers, which means that $a'_i = (\mathbf{q}'_i)^{\mathsf{T}}\boldsymbol{\pi} \in \mathbb{F}_p$. The claim then follows from completeness of Π'_{LPCP}. □

Theorem 3.7 (Soundness)**.** *If Π'_{LPCP} has soundness error (resp., strong soundness error) ε' and is bounded with respect to functions $b'_1, \ldots, b'_k$, then Construction 3.4 has soundness error (resp., strong soundness error) $\varepsilon = (k+1)/2 \cdot \varepsilon'$.*

Proof. Take any statement $\mathbf{x} \notin \mathcal{L}$ and consider an affine function $\boldsymbol{\pi}^* \in \mathbb{F}_p^\ell$ and $\delta^* \in \mathbb{F}_p$. Let $(\mathsf{st}, \mathbf{q}_{\mathsf{inp}}, \mathbf{q}) \leftarrow \mathcal{Q}_{\mathsf{LPCP}}(\tau)$, where $\mathsf{st} = (\mathsf{st}', b'_1, \ldots, b'_k, r_1, \ldots, r_k)$. Let $a_{\mathsf{inp}} = \mathbf{q}_{\mathsf{inp}}^{\mathsf{T}}\mathbf{x} \in \mathbb{F}_p$, $a^* \leftarrow \mathbf{q}^{\mathsf{T}}\boldsymbol{\pi}^* + \delta^* \in \mathbb{F}_p$ and let $\mathbf{Q}' \in \mathbb{F}_p^{\ell\times k}$ be the queries sampled by $\mathcal{Q}'_{\mathsf{LPCP}}(\tau)$ (and which are packed into $\mathbf{q}$). Let $\mathbf{q}'_1, \ldots, \mathbf{q}'_k \in \mathbb{F}_p^\ell$ be the columns of $\mathbf{Q}'$. Suppose that $\mathcal{D}_{\mathsf{LPCP}}(\mathsf{st}, a_{\mathsf{inp}}, a^*)$ outputs 1. This means that the verifier was able to extract coefficients $\mathbf{a}' = (a'_1, \ldots, a'_k)$ from a such that $\mathcal{D}_{\mathsf{LPCP}}(\mathsf{st}', a_{\mathsf{inp}}, \mathbf{a}')$ outputs 1, and moreover $a'_i \in [-b'_i, b'_i]$ for each $i \in [k]$. This means that

$$\delta^* + \sum_{i\in[k]} (\mathbf{q}'_i)^{\mathsf{T}}\boldsymbol{\pi}^* \prod_{j\in[i]} r_j = \sum_{i\in[k]} a'_i \prod_{j\in[i]} r_j \in \mathbb{F}_p,$$

or equivalently, using the fact that $r_1 = 1$,

$$f(r_2, \ldots, r_k) = (a'_1 - (\mathbf{q}'_1)^{\mathsf{T}}\boldsymbol{\pi}^* - \delta^*) + \sum_{i=2}^{k} (a'_i - (\mathbf{q}'_i)^{\mathsf{T}}\boldsymbol{\pi}^*) \prod_{j=2}^{i} r_j = 0 \in \mathbb{F}_p. \tag{3.5}$$

We consider two possibilities:

- Suppose that $a'_1 = (\mathbf{q}'_1)^{\mathsf{T}}\boldsymbol{\pi}^* + \delta^*$ and $a'_i = (\mathbf{q}'_i)^{\mathsf{T}}\boldsymbol{\pi}^*$ for $i > 1$. In this case, the adversary's response to Π'_{LPCP} is computed according to the affine strategy $\boldsymbol{\pi}^*$ and $\boldsymbol{\delta}^* = [\delta^*, 0, 0, \ldots, 0] \in \mathbb{F}_p^k$. By soundness of the Π'_{LPCP}, the probability that $\mathcal{D}_{\mathsf{LPCP}}(\mathsf{st}', \mathbf{x}, \mathbf{a}')$ accepts is at most ε' in this case.
- Suppose that either $a'_1 \neq (\mathbf{q}'_1)^{\mathsf{T}}\boldsymbol{\pi}^* + \delta^*$ or $a'_i \neq (\mathbf{q}'_i)^{\mathsf{T}}\boldsymbol{\pi}^*$ for some $i > 1$. This means that Eq. (3.5) is not the identically-zero polynomial in the variables $r_2, \ldots, r_k$. Define the polynomial $f_{a'_1,\ldots,a'_k}(r_2, \ldots, r_k)$ to be the polynomial from Eq. (3.5) for a fixed choice of $a'_1, \ldots, a'_k \in \mathbb{F}_p$. This is a polynomial of total

degree at $k-1$. By the Schwartz-Zippel lemma, the probability that $f_{a'_1,\ldots,a'_k}(r_2,\ldots,r_k)=0$ over the random choice of $r_2,\ldots r_k$ is at most

$$\Pr[f_{a'_1,\ldots,a'_k}(r_2,\ldots,r_k)=0 \mid r_2,\ldots,r_k \xleftarrow{R} [4B_{\mathsf{max}}+1, 2^{k+2}\cdot B_{\mathsf{mul}}/\varepsilon']] \leq \frac{k-1}{2^{k+2}\cdot B_{\mathsf{mul}}/\varepsilon' - 4B_{\mathsf{max}}}.$$

By a union bound, the probability that there exists $a'_1 \in [-b'_1, b'_1], \ldots, a'_k \in [-b'_k, b'_k]$ such that Eq. (3.5) holds is at most

$$\begin{aligned}\Pr[f(r_2,\ldots,r_k)=0 \mid r_2,\ldots,r_k \xleftarrow{R} [4B_{\mathsf{max}}+1, 2^{k+2}\cdot B_{\mathsf{mul}}/\varepsilon']] &\leq \frac{(k-1)\cdot\prod_{i\in[k]}(2b'_i)^k}{2^{k+2}\cdot B_{\mathsf{mul}}/\varepsilon' - 4B_{\mathsf{max}}} \\ &\leq \frac{2^k B_{\mathsf{mul}}\cdot(k-1)}{2^{k+2}\cdot B_{\mathsf{mul}}/(2\varepsilon')} = \frac{k-1}{2}\cdot\varepsilon',\end{aligned}$$

where the last inequality holds as long as $k \geq 2$, $\varepsilon' < 1$ and $b'_i \geq 9$ for all $i \in [k]$. In this case, $4B_{\mathsf{max}} \leq 2^{k+2}B_{\mathsf{mul}}/(2\varepsilon')$.

We conclude that the soundness error is bounded by $\varepsilon = \varepsilon' + (k-1)/2\cdot\varepsilon' = (k+1)/2\cdot\varepsilon'$.

Strong soundness. The proof of strong soundness follows similarly to the argument above (provided that Π'_{LPCP} satisfies strong soundness for bounded LPCPs as defined in Definition 3.3). To see this, take any statement $\mathbf{x} \in \mathbb{F}_p^n$ and consider any affine function $\boldsymbol{\pi}^* \in \mathbb{F}_p^\ell$ and $\delta^* \in \mathbb{F}_p$. We consider the probability that the verifier accepts. As before, if the verifier accepts, then Eq. (3.5) holds. We can consider the same two cases as above.

- Suppose that $a'_1 = (\mathbf{q}'_1)^\mathsf{T}\boldsymbol{\pi}^* + \delta^*$ and $a'_i = (\mathbf{q}'_i)^\mathsf{T}\boldsymbol{\pi}^*$ for $i > 1$. This is a valid affine strategy for Π'_{LPCP}, so by strong soundness of Π'_{LPCP}, the verifier here either accepts with probability 1 or with probability at most ε'.
- Suppose that either $a'_1 \neq (\mathbf{q}'_1)^\mathsf{T}\boldsymbol{\pi}^* + \delta^*$ or $a'_i \neq (\mathbf{q}'_i)^\mathsf{T}\boldsymbol{\pi}^*$ for some $i > 1$. By the above argument, the probability that the verifier accepts is at most $\varepsilon = (k-1)/2\cdot\varepsilon'$.

We conclude that Construction 3.4 has strong soundness error ε. □

Remark 3.8 (Knowledge Soundness)**.** We further note that if the underlying linear PCP Π'_{LPCP} in Construction 3.4 satisfies knowledge soundness (Definition 2.7), then the packed linear PCP from Construction 3.4 also provides knowledge soundness. Moreover, the extractor $\mathcal{E}$ for the packed linear PCP Π_{LPCP} simply runs the extractor for the underlying linear PCP Π'_{LPCP}. This follows from the fact that in the proof of Theorem 3.7, the verifier only accepts if the adversary's proof maps onto a valid strategy $\boldsymbol{\pi}^*, \delta^*$ for $\Pi_{\mathsf{LPCP}'}$. If not, then the verifier accepts with probability at most ε. This property will allow us to construct arguments of knowledge via the [BCI+13] compiler.

Theorem 3.9 (Honest-Verifier Zero Knowledge)**.** *If Π'_{LPCP} satisfies δ-HVZK then Π_{LPCP} in Construction 3.4 also satisfies δ-HVZK*

Proof. Let $\mathcal{S}'_{\mathsf{LPCP}}$ be the δ-HVZK simulator for Π'_{LPCP}. We use $\mathcal{S}'_{\mathsf{LPCP}}$ to construct a simulator $\mathcal{S}_{\mathsf{LPCP}}$ for Π_{LPCP} as follows:

1. On inputs $\mathbf{x} \in \mathbb{F}_p^n$ and τ, run $(\mathsf{st}', \mathbf{q}'_{\mathsf{inp}}, \mathbf{Q}', a'_{\mathsf{inp}}, \mathbf{a}') \leftarrow \mathcal{S}'_{\mathsf{LPCP}}(\mathbf{x}, \tau)$ where $a'_{\mathsf{inp}} \in \mathbb{F}_p$ and $\mathbf{a}' \in \mathbb{F}_p^k$.
2. Set $r_1 = 1$, sample $r_2,\ldots,r_k \xleftarrow{R} \left[4B_{\mathsf{max}}+1, B_{\mathsf{mul}}\cdot 2^{k+2}/\varepsilon'\right]$.
3. Compute the packed query $\mathbf{q} \leftarrow \sum_{i\in[k]} \mathbf{q}'_i \left(\prod_{j\in[i]} r_j\right)$, where $\mathbf{q}'_i$ denotes the i^{th} column of $\mathbf{Q}'$.
4. Compute the packed response $a \leftarrow \sum_{i\in[k]} a'_i \left(\prod_{j\in[i]} r_j\right)$.

5. Let $\mathsf{st} \leftarrow (\mathsf{st}', b'_1(\tau), \ldots, b'_k(\tau), r_1, \ldots, r_k)$ and output the tuple $(\mathsf{st}, \mathbf{q}'_{\mathsf{inp}}, \mathbf{q}, a'_{\mathsf{inp}}, a)$.

We now show that for all $\mathbf{x} \in \mathcal{L}$, the real distribution and the simulated distributions are δ-close. This follows from δ-HVZK of Π'_{LPCP}. Namely, take any $(\mathbf{x}, \mathbf{w}) \in \mathcal{R}$, and suppose we apply the above simulation procedure except we sample $(\mathsf{st}', \mathbf{q}'_{\mathsf{inp}}, \mathbf{Q}') \leftarrow \mathcal{Q}'_{\mathsf{LPCP}}(\tau)$, $\boldsymbol{\pi}' \leftarrow \mathcal{P}'_{\mathsf{LPCP}}(\tau, \mathbf{x}, \mathbf{w})$, and set $a'_{\mathsf{inp}} \leftarrow (\mathbf{q}'_{\mathsf{inp}})^{\mathsf{T}}\mathbf{x}$ and $\mathbf{a}' \leftarrow (\mathbf{Q}')^{\mathsf{T}}\boldsymbol{\pi}'$. By construction of Π_{LPCP}, the resulting tuple $(\mathsf{st}, \mathbf{q}'_{\mathsf{inp}}, \mathbf{q}, a'_{\mathsf{inp}}, a)$ is distributed identically to the real distribution where the components are generated using $\mathcal{Q}_{\mathsf{LPCP}}$ and $\mathcal{P}_{\mathsf{LPCP}}$. Next, we replace the tuple $(\mathsf{st}, \mathbf{q}'_{\mathsf{inp}}, \mathbf{Q}', a'_{\mathsf{inp}}, \mathbf{a}')$ generated by $\mathcal{Q}'_{\mathsf{LPCP}}$ and $\mathcal{P}'_{\mathsf{LPCP}}$ with the output of $\mathcal{S}'_{\mathsf{LPCP}}(\mathbf{x}, \tau)$. By δ-HVZK of Π'_{LPCP}, these two distributions are δ-close. But this is precisely the distribution output by $\mathcal{S}_{\mathsf{LPCP}}$, and the claim follows by a standard hybrid argument. □

Corollary 3.10 (Linear PCP Packing)**.** *Let Π'_{LPCP} be a k-query bounded LPCP over $\mathbb{F}_p$ for a binary relation $\mathcal{R}$ with bound functions $b'_1, \ldots, b'_k$, soundness error (resp., strong soundness error) ε', and which satisfies δ-HVZK. Let $B'_{\mathsf{min}} = \min_{i \in [k]} b'_i$, $B'_{\mathsf{mul}} = \prod_{i \in [k]} b'_i$, and $B = 2B'_{\mathsf{min}} \left(B'_{\mathsf{mul}} 2^{k+2}/\varepsilon\right)^{k-1}$. If $p > 2B$, then there exists a 1-query bounded LPCP over $\mathbb{F}_p$ for $\mathcal{R}$ with bound B, soundness error (resp., strong soundness error) $(k+1)/2\varepsilon'$, and which satisfies δ-HVZK.*

3.1 Constructing 1-Query Bounded Linear PCPs

As noted in [BCI+13], a simple extension of the Hadamard PCP from [ALM+98, IKO07] to arbitrary finite fields $\mathbb{F}$ yields a 3-query linear PCP for arithmetic circuit satisfiability over $\mathbb{F}$. We review the construction in Appendix B.1. There, we make two additional observations:

- The 3-query Hadamard linear PCP decision procedure is in fact linear in two of the responses, so we can combine two of the queries together to obtain a 2-query linear PCP over $\mathbb{F}$ (see Construction B.1).
- For an arithmetic circuit C of size s over a finite field $\mathbb{F}$, an (honest) proof $\boldsymbol{\pi}$ for a statement-instance pair $(\mathbf{x}, \mathbf{w})$ consists of a collection of wire labels $\mathbf{z} \in \mathbb{F}^s$ corresponding to $C(\mathbf{x}, \mathbf{w})$, together with the pairwise products $\mathbf{z} \otimes \mathbf{z}$. If we are working with Boolean circuits (or more generally, arithmetic circuits over the integers where the wire labels are bounded; see Remark 3.16), then the resulting linear PCP is in fact a bounded linear PCP (see Construction B.8). We state this formally below.

Theorem 3.11 (2-Query Bounded LPCP)**.** *Let $C \colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ be a Boolean circuit of size s, and let $\mathbb{F}_p$ be a finite field. Let $\tau \in \mathbb{N}$ be a bound parameter. There exists a 2-query bounded linear PCP Π_{LPCP} over $\mathbb{F}_p$ with algebraic degree 2 for the relation $\mathcal{R}_C$ with perfect completeness, strong soundness error $2/\tau$, and query length $(s^2+3s)/2$. Moreover,*

- *Π_{LPCP} is bounded with respect to bound functions $b_1(\tau) = s\tau/2$ and $b_2(\tau) = 2(b_1(\tau))^2$; and*
- *for any $\varepsilon > 0$, Π_{LPCP} is bounded with respect to functions $b'_1(\tau) = \tau\sqrt{s/2 \cdot \ln(2/\varepsilon)}$ and $b'_2(\tau) = 2(b'_1(\tau))^2$ with probability $1 - \varepsilon$.*

Furthermore, the query-generation algorithm $\mathcal{Q}_{\mathsf{LPCP}}$ and the prover algorithm $\mathcal{P}_{\mathsf{LPCP}}$ runs in time $O(s^2) \cdot \mathsf{polylog}(p)$. The decision algorithm $\mathcal{D}_{\mathsf{LPCP}}$ runs in time $\mathsf{polylog}(p)$.[12]

The construction in Theorem 3.11 is a simple adaptation of the basic Hadamard linear PCP and we provide the description (Construction B.8) and analysis in Appendix B.2.

[12] Recall that in our setting, the verification algorithm takes as input a linear function of the statement $x \in \{0,1\}^n$, and not the statement x itself as input.

Bounded LPCPs with zero-knowledge. The bounded linear PCP in Theorem 3.11 is not zero-knowledge. However, as noted in [BCI+13], it is straightforward to augment the Hadamard linear PCP to provide honest-verifier zero knowledge (by introducing a "dummy wire;" see Remark B.5). We can leverage the same idea in our setting to obtain a bounded linear PCP that provides δ-honest-verifier zero knowledge. We state the main theorem here, but refer to Appendix B.3 for the full description (Construction B.13) and analysis. Note that this construction does *not* provide strong soundness (Remark B.17).

Theorem 3.12 (2-Query δ-HVZK Bounded LPCP)**.** *Let $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ be a Boolean circuit of size s, and let $\mathbb{F}_p$ be a finite field. Let $\tau \in \mathbb{N}$ be a bound parameter and $\delta > 0$ be a zero-knowledge parameter. There exists a 2-query bounded linear PCP Π_{LPCP} with algebraic degree 2 over $\mathbb{F}_p$ for the relation $\mathcal{R}_C$ with perfect completeness, soundness error $2/\tau$, δ-honest-verifier zero knowledge, and query length $(s^2+3s)/2$. Moreover,*

- *Π_{LPCP} is bounded with respect to bound functions $b_1(\tau) = s\tau/2 + 2\tau\sqrt{s/2\ln(4/\delta)}/\delta$ and $b_2(\tau) = 2(b_1(\tau))^2$.*
- *for any $\varepsilon > 0$, Π_{LPCP} is bounded with respect to $b'_1(\tau) = \tau\sqrt{s/2}\big(\sqrt{\ln(2/\varepsilon)} + 2/\delta\sqrt{\ln(4/\delta)}\big)$ and $b'_2(\tau) = 2(b_1(\tau))^2$ with probability at least $1-\varepsilon$.*

Furthermore, the query-generation algorithm $\mathcal{Q}_{\mathsf{LPCP}}$ and the prover algorithm $\mathcal{P}_{\mathsf{LPCP}}$ runs in time $O(s^2) \cdot \mathsf{polylog}(p)$. The decision algorithm $\mathcal{D}_{\mathsf{LPCP}}$ can be implemented by an arithmetic circuit of size $\mathsf{polylog}(p)$.

Remark 3.13 (Comparison with Square Span Programs [DFGK14])**.** As described in [Gro16], one can also construct a 2-query linear PCP (a "non-interactive linear proof (NILP)") from the square span programs of [DFGK14]. In particular, starting from a square span program, it is possible to construct a 2-query linear PCP for arithmetic circuit satisfiability over a field $\mathbb{F}$ with query length is $O(s)$ and soundness error $O(s/|\mathbb{F}|)$, where s is the circuit size. Like the 3-query linear PCPs based on quadratic span programs [GGPR13], the advantage here is that the query length is $O(s)$ rather than $O(s^2)$ as in the Hadamard construction. However, these linear PCPs are not bounded: the verifier's query is constructed by computing the powers of a random field element $\alpha \in \mathbb{F}$: $\alpha, \alpha^2, \ldots, \alpha^s$. As such, even if α is sampled from a bounded range, the absolute magnitude of some of the components will be comparable to $|\mathbb{F}|$. If we then consider the magnitude of the responses computed over the integers, they will not lie in an interval smaller than $|\mathbb{F}|$. It is not clear how to apply our linear PCP packing transformation to linear PCPs based on square span programs or quadratic span programs. It is an interesting open question to construct a 1-query linear PCP based on square span programs or quadratic span programs (or more generally, any 1-query linear PCP with query length $O(s)$) that satisfies strong soundness.

1-query bounded LPCPs. We can now combine our 2-query bounded linear PCPs (Theorems 3.11 and 3.12) with our linear PCP packing construction (Corollary 3.10) to obtain a 1-query linear PCP for Boolean circuit satisfiability. We summarize our construction in the corollary below:

Corollary 3.14 (1-Query Bounded LPCP)**.** *Let $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ be a Boolean circuit of size s, and let $\mathbb{F}_p$ be a finite field. Let $\tau \in \mathbb{N}$ be a bound parameter and $\delta > 0$ be a zero-knowledge parameter. There exist 1-query bounded linear PCPs over $\mathbb{F}_p$ for $\mathcal{R}_C$ with the following properties:*

- *Perfect completeness, strong soundness error $3/\tau$, query length $(s^2+3s)/2$, and bound function $b(\tau) = 4s^4\tau^5/3$, provided that $p > 2b(\tau)$; and*
- *Perfect completeness, soundness error $3/\tau$, query length $(s^2+3s)/2$, δ-honest-verifier zero knowledge, and bound function $b(\tau,\delta) = 64\tau(s\tau/2 + 2\tau\sqrt{s/2\ln(4/\delta)}/\delta)^4/3$, provided that $p > 2b(\tau,\delta)$.*

Additionally, the query-generation algorithm $\mathcal{Q}_{\mathsf{LPCP}}$ and the prover algorithm $\mathcal{P}_{\mathsf{LPCP}}$ runs in time $O(s^2) \cdot \mathsf{polylog}(p)$. The verifier's decision algorithm $\mathcal{D}_{\mathsf{LPCP}}$ runs in time $\mathsf{polylog}(p)$.

Remark 3.15 (Field Size Bound in Corollary 3.14)**.** The packing construction from Corollary 3.14 imposes a lower bound on the field size. If we are working over a standard 256-bit group, this translates to a constraint on the circuit size, soundness error, and statistical zero-knowledge parameter that we are able to support. However in concrete settings, these bounds are unlikely to be the bottleneck. For instance, if $\log p \approx 256$, we can support circuits with size 2^{25} while achieving a soundness error of 2^{-28} and providing δ-HVZK for $\delta = 10^{-3}$. These values are beyond the range of parameters for which the resulting SNARG construction is competitive (see Table 2).

Remark 3.16 (Arithmetic Circuits over the Integers)**.** As noted in Construction B.8, our basic constructions naturally extend to support general arithmetic circuits over the integers, provided that each of the wire values are small and moreover, the output wire of each gate in the circuit can be expressed as a quadratic polynomial in the input wires to the gate. This includes gates that can take *many* input values, as long as the output value is bounded and corresponds to a quadratic function of the input values. For example, we can consider circuits that have large fan-in "inner product" gates which compute an inner product between the two input vectors. The size of the resulting linear PCP then scales quadratically with the number of gates (and input wires) in the circuit. By compressing large computations like inner products into a *single* gate operation, the resulting linear PCP can be substantially shorter than that obtained by first translating the computation into an equivalent Boolean circuit and constructing the linear PCP from the circuit. Thus, for structured computations like matrix multiplication, it is possible to significantly reduce the linear PCP size by relying on these "Hadamard-friendly" gates. It is an interesting challenge to construct other Hadamard-friendly gadgets. For instance, if we can construct a cryptographic hash function from such gadgets, that would give an efficient linear PCP, and thus, a succinct argument, for verifying the output of a hash function.

Remark 3.17 (Strong Soundness for 1-Query Linear PCPs)**.** An appealing property of the 1-query linear PCP from Corollary 3.14 (without zero-knowledge) is that it provides *strong soundness*. As shown in [BCI+13], linear PCPs with strong soundness are a useful ingredient for constructing *multi-theorem* succinct arguments. Our construction in Corollary 3.14 is the first 1-query linear PCP that provides strong soundness. In contrast, the linear PCP based on classic PCPs from [BCI+13] cannot provide strong soundness, and as they show, any linear PCP constructed by packing together queries to a classic PCP cannot satisfy strong soundness.

3.2 SNARGs based on ElGamal

In this section, we describe how to efficiently compile our 1-query bounded linear PCP from Corollary 3.14 to obtain a designated-verifier SNARG in the preprocessing model where the proof size consists of 2 group elements and where the verification complexity is sublinear in the cost of the classic NP verifier. While it is possible to directly invoke the [BCI+13] compiler on our 1-query bounded linear PCP together with ElGamal encryption, the verification complexity of the resulting scheme is *quadratic* in the size of the classic NP verifier. This is because the additively-homomorphic version of ElGamal encryption scheme (see Construction C.4) encodes the messages in the exponent, and decryption requires solving a discrete log. When the responses are bounded in an interval of size B, this can be done in time $O(\sqrt{B})$ using generic algorithms (e.g., [Pol78]). While a bounded linear PCP seems like a natural choice to use in conjunction with ElGamal, the bounds in Corollary 3.14 scale with s^4, where s is the circuit size of the classic NP verifier. Instantiating with ElGamal then yields an unacceptable verification complexity that is quadratic in the circuit size. In this section, we will leverage the structure of our packed 2-query bounded linear PCP to obtain an asymptotically-faster (worst-case verification complexity that scales with $\widetilde{O}(\sqrt{s})$) and concretely-efficient designated-verifier SNARG based on ElGamal. Compared with a direct instantiation of the [BCI+13] compiler with ElGamal encryption, our proofs are either 4 times more succinct (2 group elements vs. 8 group elements) or much more concretely efficient (relying on linear PCPs rather than classic PCPs).

The ElGamal instantiation. Before describing how we optimize the verification procedure for our ElGamal-based SNARG, we begin with an explicit description of the construction. We assume a 1-query bounded linear PCP, in which case we obtain a *direct* construction from any linear-only encryption scheme:

Construction 3.18 (SNARG based on ElGamal)**.** Let $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ be a Boolean circuit of size s. Let $\mathsf{GroupGen}$ be a prime-order group generator (Definition 2.3) that outputs a group $\mathbb{G}$ of prime order p. Let $\Pi_{\mathsf{LPCP}} = (\mathcal{Q}_{\mathsf{LPCP}}, \mathcal{P}_{\mathsf{LPCP}}, \mathcal{D}_{\mathsf{LPCP}})$ be a 1-query bounded linear PCP with bound $B = B(\tau)$ and bound parameter τ for the relation $\mathcal{R}_C$ over $\mathbb{F}_p$. We construct a SNARG $\Pi_{\mathsf{SNARG}} = (\mathcal{S}_{\mathsf{SNARG}}, \mathcal{P}_{\mathsf{SNARG}}, \mathcal{V}_{\mathsf{SNARG}})$

- $\mathcal{S}_{\mathsf{SNARG}}(1^\lambda)$: On input the security parameter λ, the setup algorithm samples a group description $(\mathbb{G}, p, g) \leftarrow \mathsf{GroupGen}(1^\lambda)$ and a linear PCP query $(\mathsf{st}', \mathbf{q}_{\mathsf{inp}}, \mathbf{q}) \leftarrow \mathcal{Q}_{\mathsf{LPCP}}(\tau) \in \mathbb{F}_p^\ell$. The setup algorithm samples a secret key $\alpha \xleftarrow{\mathrm{R}} \mathbb{F}_p$, and computes the ElGamal public key $h \leftarrow g^\alpha$. The setup algorithm samples $\mathbf{r} \xleftarrow{\mathrm{R}} \mathbb{F}_p^\ell$, and computes the ciphertexts $(g^{\mathbf{r}}, h^{\mathbf{r}} g^{\mathbf{q}})$, where for a vector $\mathbf{r}$, we write $g^{\mathbf{r}}$ to denote the vector of group elements $(g^{r_1}, \ldots, g^{r_\ell})$. The setup algorithm outputs the common reference string crs and verification state st:
$$\mathsf{crs} = \big((\mathbb{G}, p, g), h, (g^{\mathbf{r}}, h^{\mathbf{r}} g^{\mathbf{q}}), \tau\big) \text{ and } \mathsf{st} = (\mathsf{st}', \alpha).$$
- $\mathcal{P}_{\mathsf{SNARG}}(\mathsf{crs}, x, w)$: On input a common reference string $\mathsf{crs} = \big((\mathbb{G}, p, g), h, (g^{\mathbf{r}}, g^{\mathbf{s}}), \tau\big)$, a statement $x \in \{0,1\}^\ell$, and a witness $w \in \{0,1\}^h$, the prover algorithm computes a proof $\boldsymbol{\pi} \leftarrow \mathcal{P}_{\mathsf{LPCP}}(\tau, x, w) \in \mathbb{F}_p^\ell$. It computes the proof as $\pi = (g^{\mathbf{r}^\mathsf{T}\boldsymbol{\pi}}, g^{\mathbf{s}^\mathsf{T}\boldsymbol{\pi}})$.
- $\mathcal{V}_{\mathsf{SNARG}}(\mathsf{st}, x, \pi)$: On input the verification state $\mathsf{st} = (\mathsf{st}', \mathbf{q}_{\mathsf{inp}}, \alpha)$, the statement $x \in \{0,1\}^n$, and a proof $\pi = (g_1, g_2)$, the verifier computes $h' = g_2/g_1^\alpha$, and checks if there exists $a \in [-B, B]$ such that $h' = g^a$. It outputs $\mathcal{D}_{\mathsf{LPCP}}(\mathsf{st}', \mathbf{q}_{\mathsf{inp}}^\mathsf{T} x, a)$.

Assuming that the ElGamal encryption scheme satisfies linear targeted malleability with respect to the target set $[-B, B]$ (Definition A.6), then Construction 3.18 is a designated-verifier SNARG in the preprocessing model where the proof size consists of exactly 2 group elements (via Theorem A.9). The bottleneck is the expensive verification procedure. As stated, the verification algorithm has to compute the discrete log of $h' = g^a$ where a is the linear PCP response. This can be computed in time $\widetilde{O}(\sqrt{B})$, which for the linear PCP from Corollary 3.14, is *quadratic* in the circuit size s.

Optimizing the verification procedure. We now describe how to more efficiently implement the verification procedure in Construction 3.18 to obtain a SNARG whose worst-case verification complexity is $O(s)$. Moreover, if we allow for a negligible completeness error (as opposed to *perfect* completeness), we give a procedure whose worst-case verification complexity is $\widetilde{O}(\sqrt{s})$. Our optimization will rely on specific properties of the linear PCP from Corollary 3.14. Namely, the 1-query linear PCP from Corollary 3.14 was obtained by packing together a 2-query linear PCP (Theorems 3.11 and 3.12). For simplicity, we will describe our optimization for the construction based on Theorem 3.11 (Construction B.8), but the same techniques apply to the construction from Theorem 3.12 (Construction B.13).

- Construction B.8 has the property that the verifier accepts a response $(a_1, a_2) \in \mathbb{F}_p^2$ only if $a_1^2 + a_2 = a_{\mathsf{inp}} + u_C$, where $a_{\mathsf{inp}}, u_C \in \mathbb{F}_p$ are scalars that are known to the verifier.
- If $\mathbf{q}_1, \mathbf{q}_2 \in \mathbb{F}_p^\ell$ are the two queries the verifier makes in Construction B.8, then the packed query in Construction 3.4 satisfies $\mathbf{q} = \mathbf{q}_1 + r \cdot \mathbf{q}_2$ for some $r \in \mathbb{F}_p$ known to the verifier. This means that the honest prover's response satisfies $a = \mathbf{q}^\mathsf{T}\boldsymbol{\pi} = \mathbf{q}_1^\mathsf{T}\boldsymbol{\pi} + r \cdot \mathbf{q}_2^\mathsf{T}\boldsymbol{\pi} = a_1 + r \cdot a_2$.
- Finally, Construction B.8 has the property that for an accepting proof, the first response a_1 satisfies $a_1 \in [-b_1(\tau), b_1(\tau)] = [-s\tau/2, s\tau/2]$.

Equivalently, this means the linear PCP verifier in Construction B.8 accepts only if there exists $a_1 \in [-b_1(\tau), b_1(\tau)]$ such that the following two relations hold:
$$a_2 = a_{\mathsf{inp}} + u_C - a_1^2 \text{ and } a = a_1 + r \cdot a_2,$$

or equivalently, if there exists $a_1 \in [-b_1(\tau), b_1(\tau)]$ such that

$$a = a_1 + r \cdot (a_{\mathsf{inp}} + u_C - a_1^2).$$

In the SNARG from Construction 3.18, the verifier first computes g^a. Now, instead of recovering a by solving discrete log, the verifier instead checks whether there exists $a_1 \in [-b_1(\tau), b_1(\tau)]$ where

$$g^a = g^{a_1 + r(a_{\mathsf{inp}} + u_C - a_1^2)}. \tag{3.6}$$

This can be done by performing a brute force search over all of the possible $2b_1(\tau)$ values for a_1 and seeing if Eq. (3.6) holds. We can also rewrite Eq. (3.6) as checking whether there exists a_1 where

$$g^a \cdot g^{-r(a_{\mathsf{inp}} + u_C)} = g^{a_1 - ra_1^2}. \tag{3.7}$$

Observe now that the right-hand side of the expression depends only on the value of a_1 and r, and in particular, is *independent* of the statement. Since r is sampled by the setup algorithm $\mathcal{S}_{\mathsf{SNARG}}$, the verifier can actually *precompute* a table of values $g^{a_1 - ra_1^2}$ for each possible value of a_1. Then, to verify a proof $\pi = (g_1, g_2)$, the verifier computes $u = g^a g^{-r(a_{\mathsf{inp}} + u_C)}$, which requires a *constant* number of group operations, and finally, checks to see whether u is contained in the table or not. This yields a substantially faster verification procedure. Even without this optimization, we obtain a SNARG where the verification complexity is $O(s)$. We summarize this in the following corollary:

Corollary 3.19 (SNARG from ElGamal with Perfect Completeness)**.** *Let $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ be a Boolean circuit of size s. Let $\varepsilon > 0$ be a soundness parameter and $\delta > 0$ be a zero-knowledge parameter. Assuming the ElGamal encryption scheme (over a prime order group $\mathbb{G}$ of order p) satisfies linear targeted malleability with respect to a target message space $[-B, B]$ for $B = \mathsf{poly}(s, 1/\varepsilon, 1/\delta)$, there exist a designated-verifier SNARGs for $\mathcal{R}_C$ with perfect completeness, non-adaptive soundness error ε, and proofs of size $2\log|\mathbb{G}|$. The CRS has size $O(s^2)$. The setup algorithm and prover run in time $O(s^2)$ and the verifier runs in time $O(s/\varepsilon)$. Moreover, the SNARG can be extended to satisfy δ-HVZK. In this case, the verifier runs in time $O(\sqrt{s}/\varepsilon \cdot (\sqrt{s} + \sqrt{\log(1/\delta)/\delta}))$; the setup and prover complexity remain unchanged. All of the running times are up to $\mathsf{polylog}(p)$ factors.*

Sublinear verification. We can further reduce the verification complexity by having the verifier only *accept* proofs where the first response a_1 is contained in a much shorter interval. Instantiating Theorem 3.11 with any $\varepsilon = \lambda^{-\omega(1)}$, we have that the bound $b_1'(\tau) = \widetilde{O}(\tau\sqrt{s})$ with probability $1 - \varepsilon = 1 - \mathsf{negl}(\lambda)$. We now modify the verification procedure $\mathcal{V}_{\mathsf{SNARG}}$ to only accept if there exists $a_1 \in [-b_1'(\tau), b_1'(\tau)]$ such that Eq. (3.7) holds. Since the subset of proofs the verifier accepts is now a strict *subset* of the proofs it accepted in the original scheme, (single-theorem) soundness is preserved. The trade-off is that the verifier may now reject some honestly-generated proofs, but Theorem 3.11 says this can only happen with negligible probability over the verifier's randomness. This yields the following theorem:

Corollary 3.20 (SNARG from ElGamal with Sublinear Verification)**.** *Let $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ be a Boolean circuit of size s. Let $\varepsilon > 0$ be a soundness parameter and $\delta > 0$ be a zero-knowledge parameter. Assuming the ElGamal encryption scheme (over a prime order group $\mathbb{G}$ of order p) satisfies linear targeted malleability with respect to a target message space $[-B, B]$ for $B = \mathsf{poly}(s, 1/\varepsilon, 1/\delta)$, there exists a designated-verifier SNARG for $\mathcal{R}_C$ with* statistical *completeness and non-adaptive soundness error ε, and proofs of size $2\log|\mathbb{G}|$. The CRS has size $O(s^2)$. The setup algorithm and prover run in time $O(s^2)$ and the verifier runs in time $\widetilde{O}(\sqrt{s}/\varepsilon)$. Moreover, the SNARG can be extended to satisfy δ-HVZK. In this case, the verifier runs in time $\widetilde{O}(\sqrt{s\log(1/\delta)}/(\delta\varepsilon))$; the setup and prover complexity remain unchanged. All of the running times are up to $\mathsf{polylog}(p)$ factors.*

A preprocessing variant. As mentioned above, the verification relation in Eq. (3.7) is very amenable to preprocessing. Namely, the verifier can perform a one-time setup and precompute all of the accepting

values of $g^{a_1 - r a_1^2}$ and store them in a table. Applying the sublinear verification approach described above, this will only require $\widetilde{O}(\sqrt{s})$ space when verifying circuits of size s. In the online phase, to check a proof, the verifier needs to perform a single ElGamal decryption, followed by computing the left-hand side of Eq. (3.7), and finally, a single table lookup. The overall computation comes out to just 2 exponentiations and 2 multiplications, followed by the table lookup. This yields the following corollary:

Corollary 3.21 (SNARG from ElGamal with Preprocessing)**.** *Let $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ be a Boolean circuit of size s. Let $\varepsilon > 0$ be a soundness parameter and $\delta > 0$ be a zero-knowledge parameter. Assuming the ElGamal encryption scheme (over a prime order group $\mathbb{G}$ of order p) satisfies linear targeted malleability with respect to a target message space $[-B, B]$ for $B = \mathsf{poly}(s, 1/\varepsilon, 1/\delta)$, there exists a designated-verifier SNARG for $\mathcal{R}_C$ with* statistical *completeness, non-adaptive soundness error ε, and proofs of size $2\log|\mathbb{G}|$. The CRS has size $O(s^2)$. The setup algorithm runs in time $\widetilde{O}(s^2 + \sqrt{s}/\varepsilon)$ and outputs a table T of size $\widetilde{O}(\sqrt{s}/\varepsilon)$. The prover runs in time $O(s^2)$ and the verifier runs in time $\widetilde{O}(1)$ given access to the precomputed table T. If we extend the SNARG to provide δ-HVZK, the setup algorithm now runs in time $O(s^2 + \sqrt{s\log(1/\delta)}/(\delta\varepsilon))$ and outputs a table T of size $O(\sqrt{s\log(1/\delta)}/(\delta\varepsilon))$. Given access to the precomputed table, the verifier's runtime is $\widetilde{O}(1)$. All of the running times and table sizes are up to $\mathsf{polylog}(p)$ factors.*

Remark 3.22 (Arguments of Knowledge)**.** The SNARG constructions in Corollaries 3.19 to 3.21 are all arguments of knowledge (i.e., "SNARKs") since the underlying linear PCPs from Appendix B provide knowledge soundness and Construction 3.4 preserves the knowledge soundness of the underlying linear PCP.

Remark 3.23 (Reducing the Table Size)**.** For many parameter settings (see Section 3.3 and Table 2), the size of the lookup table required to implement fast verification is modest. However, in cases where space is a premium, there are several ways to reduce the space at the expense of other parameters. One approach is to trade-off space usage for soundness by storing the elements in a bloom filter [Blo70] instead of a hash-table. Another approach that trades off space for verification time is to construct a Hellman table [Hel80] for inverting the function $x \mapsto g^{x - r x^2}$, which precisely corresponds to the verification relation (note that the scaling factor r is known at preprocessing time). In this case, if there are $\widetilde{O}(s^{1/2})$ possible values for x, a Hellman table would enable inversion in time roughly $\widetilde{O}(s^{1/3})$ and require a table of size $\widetilde{O}(s^{1/3})$.

Remark 3.24 (Adaptive Soundness)**.** All of the SNARG constructions in Corollaries 3.19 to 3.21 satisfy *adaptive* soundness (where the prover can choose the statement *after* seeing the CRS) if we model the group $\mathbb{G}$ of the ElGaml encryption scheme as a generic group (see Remark A.10 and Theorem C.10).[13] In this setting of adaptively-sound SNARGs, the Gentry-Wichs lower bound [GW11] rules out the possibility of basing security on a falsifiable assumption, on some non-falsifiable assumption or working in an idealized model seems necessary in the security analysis.

An alternative instantiation from Paillier. We can also compile our 1-query (bounded) linear PCP from Corollary 3.14 to obtain a designated-verifier SNARG using an encryption scheme based on Paillier. We discuss this below.

Remark 3.25 (Paillier-Based Instantiation)**.** Assuming that the Paillier encryption scheme satisfies linear targeted malleability, we can directly apply the [BCI+13] compiler to obtain a SNARG where the proof consists of a *single* Paillier ciphertext. Moreover, if we choose the bound parameter $\tau = \lambda^{\omega(1)}$ to be super-polynomial in the security parameter λ, then the soundness error of the linear PCP from Corollary 3.14 becomes negligible, and we obtain a designated-verifier SNARG with negligible soundness error based on Paillier. Previously, Bitansky et al. [BCI+13] constructed a SNARG based on Paillier with the same level of succinctness (a single group element) by starting from a single-query linear PCP based on a classic PCP. We compare our instantiations here:

[13] Bitansky et al. [BCI+13] also showed that if the underlying encryption scheme satisfies a stronger extractability notion called "linear-only" security, then the resulting SNARG also satisfies adaptive soundness. However, it is not clear that the vanilla ElGamal encryption scheme satisfies this stronger extractability notion (even if we work in the generic group model).

- **Reusability:** One of our advantage of our 1-query linear PCP (without zero-knowledge) is that it provides *strong* soundness. This is not true of the 1-query linear PCP from [BCI+13] (see Remark 3.17). As such, if we make a stronger interactive linear-only assumption on Paillier, we obtain a *reusable* designated-verifier SNARG from the Paillier assumption where the proof consists of just two Paillier ciphertexts. No such reusable Paillier-based construction with this level of succinctness is known from [BCI+13].
- **Linear PCPs vs. classic PCPs:** The 1-query linear PCP from [BCI+13] relied on *classic* PCP as a starting point (i.e., the prover must first encode its statement and witness using a classic PCP). This incurs a substantial concrete overhead on the efficiency of the resulting construction, especially compared to more lightweight linear PCPs based on the Hadamard code or quadratic span programs [GGPR13]. In contrast, our construction shows how to pack *linear PCPs* into a 1-query linear PCP.
- **CRS size and prover complexity:** One advantage of the [BCI+13] constructions is that the size of the CRS, and correspondingly, the prover complexity is quasi-linear in the circuit size, while in our construction, both the CRS and the prover complexity are quadratic in the circuit size (due to our reliance on the Hadamard linear PCP for our underlying packing construction).

3.3 Concrete Efficiency of the ElGamal-Based SNARG

In this section, we provide some concrete performance estimates for the designated-verifier SNARG from Corollaries 3.20 and 3.21 for different circuit sizes, soundness parameters, and zero-knowledge parameters. We estimate the size of the CRS (based on the number of group elements), the prover complexity (in terms of the number of elementary group operations), and the verifier's cost (measured in the size of its lookup table in the case of the preprocessing construction from Corollary 3.21 and its time complexity in the case of the non-preprocessing variant from Corollary 3.20). The main results are summarized in Table 2.

CRS size. From Theorem 3.11, the query length of the linear PCP for a Boolean circuit of size s is $(s^2 + 3s)/2$. In the SNARG, each element in the CRS is encrypted with ElGamal encryption. While a standard ElGamal ciphertext consist of two group elements, we note that the first component is uniformly random and *independent* of the message. Thus, rather than include it as part of the CRS, it can instead be derived as the output of a random oracle. In this case, the CRS only needs to contain the message-embedding component (i.e., the second component) of each ElGamal ciphertext. Note that in this setting where the first half of the ElGamal ciphertext is derived from the output of a random oracle, knowledge of the ElGamal secret key is necessary to construct the ciphertexts. By appealing to the random oracle heuristic, the CRS only needs to contain one group element for each component of the linear PCP query vector. Each group element can be represented by $\log p$ bits, thus yielding an overall CRS size of $(s^2 + 3s)/2 \cdot \log p$ bits.

Prover complexity. The prover has to homomorphically compute the inner product between the encrypted query in the CRS and its proof vector $\boldsymbol{\pi}$. Recall that in the Hadamard linear PCP (Appendix B), an honestly-generated proof vector $\boldsymbol{\pi} = [\mathbf{z}, \mathbf{z} \otimes \mathbf{z}]$ consists of the wire labels $\mathbf{z}$ to the Boolean circuit together with the pairwise products $\mathbf{z} \otimes \mathbf{z}$. If we assume that the wire values $\mathbf{z}$ to the Boolean circuit to be roughly split between 0 and 1, then we would expect roughly 3/4 of the entries in $\boldsymbol{\pi}$ to be 0. For each non-zero entry in the proof vector, the prover has to perform two group operations, so the total number of group operations the prover performs when the wire labels to the circuit are roughly balanced is $(s^2 + 3s)/4 \cdot \log p$. In the worst case where most of the wire labels to the circuit are 1, the prover would have to perform $(s^2 + 3s) \cdot \log p$ group operations.

Verifier complexity. For our concrete efficiency estimates, we focus on the constructions with sublinear verification complexity (Corollaries 3.20 and 3.21). Let c be the desired completeness error, ε be the desired soundness error, and δ be the desired honest-verifier zero-knowledge parameter. The verification complexity is dominated by the cost of checking whether there exists a value $a_1 \in [-b_1(\tau), b_1(\tau)]$ that satisfies Eq. (3.7),

where b_1 denotes the bound on the response to the first query in the underlying linear PCP, and τ is the bound parameter for the bounded linear PCP. Let N be the size of this interval. For our bounded linear PCP based on the Hadamard code (Appendix B), if we set the completeness error to c, then by Theorem B.18, we have that

$$b_1(\tau) = \tau\sqrt{s/2}\left(\sqrt{\ln(2/c)} + 2/\delta\sqrt{\ln(4/\delta)}\right). \tag{3.8}$$

By Corollary 3.14, the SNARG obtained from this linear PCP satisfies soundness error $\varepsilon = 3/\tau$. Thus, we set $\tau = 3/\varepsilon$ in Eq. (3.8) to achieve the desired level of soundness. This means that

$$N = 2b_1(3/\varepsilon) = 6/\varepsilon \cdot \sqrt{s/2} \cdot \left(\sqrt{\ln(2/c)} + 2/\delta\sqrt{\ln(4/\delta)}\right).$$

We consider two settings, depending on whether the verifier has a lookup table or not:

- **Verifier storage with lookup table:** The lookup table needs to store an entry for each of the N values. To reduce space, instead of storing the full group element ($\log p$ bits), we instead store a hash of the element. If the output of the hash function is $2\log N$ bits, then with constant probability, there will not be a collision in their hash values (we can also ensure this by repeatedly sampling hash functions until finding one without collisions). Along with each hash value, we store the value of $a_1 \in [-b_1(\tau), b_1(\tau)]$ to which it corresponds. Overall, the lookup table requires $3\log N$ bits of storage.
- **Verifier complexity without lookup table:** In the setting without a lookup table, the verifier can exhaustively search the interval $[-b_1(\tau), b_1(\tau)]$ with a total of $2N$ group multiplications. In particular, the verifier leverages the fact that in Eq. (3.7),

  $$g^{(a_1+1)-r(a_1+1)^2} = g^{(a_1-ra_1^2)}g^{(1-2a_1r)}.$$

  Thus, if the verifier has computed values $g^{a_1-ra_1^2}$ and g^{1-2a_1r}, it can compute the values $g^{(a_1+1)-r(a_1+1)^2}$ $g^{1-2(a_1+1)r} = g^{1-2a_1r}g^{-2r}$ with two group multiplications.

Microbenchmarks for group operations. To provide estimates for the concrete running time of our SNARG, we measure running times for a single group exponentiation and a single group operations (averaged over 100,000 runs) on the Curve25519 elliptic curve [Ber06] using the `libsodium` implementation on a standard laptop. Based on our estimates, a single group exponentiation requires ≈ 0.054ms of computation while a single group multiplication requires ≈ 0.014ms. Each group element is represented by 32 bytes.

4 1-Query Linear PCP from Hardness of Approximation

In this section, we show how to construct a 1-query *instance-dependent* linear PCP with a linear decision procedure and *negligible* soundness error. Combined with the compiler from [BCI+13], and assuming linear targeted malleability of ElGamal encryption, we obtain the first laconic argument with negligible soundness error where the proof consists of a *single* ElGamal ciphertext. Note that we do *not* obtain a SNARG because the verifier's first message (i.e., the verifier's query) depends on the statement.

The minimum weight solution problem. The basis of our new 1-query linear PCP is the promise version of of the minimum weight solution problem (MWSP) from [KPV12, Definition 2.4]. While [KPV12] defines the problem with respect to the binary field $\mathbb{F}_2$, we consider a generalization of the problem to general finite fields.

Definition 4.1 (Gap Minimum Weight Solution Problem (GapMWSP))**.** For an approximation factor $\beta > 1$, an instance of $\mathsf{GapMWSP}_\beta$ is a triple $(\mathbf{A}, \mathbf{b}, d)$ where $\mathbf{A} \in \mathbb{F}^{\ell\times n}$, $\mathbf{b} \in \mathbb{F}^\ell$ for some finite field $\mathbb{F}$, and $d \in \mathbb{N}$ such that

- $(\mathbf{A}, \mathbf{b}, d)$ is a YES instance if there exists a solution $\mathbf{x} \in \mathbb{F}^n$ such that $\mathbf{Ax} = \mathbf{b}$ and $\mathbf{v}$ has Hamming weight at most d (i.e., $\mathbf{v}$ has at most d nonzero entries).
- $(\mathbf{A}, \mathbf{b}, d)$ is a NO instance if every solution $\mathbf{x} \in \mathbb{F}^n$ where $\mathbf{Ax} = \mathbf{b}$ has Hamming weight at least $\beta \cdot d$.

Lemma 4.2 (Hardness of GapMWSP)**.** *For any constant $c > 0$, $\beta = \log^c n$ and any finite field $\mathbb{F}$ where $\log|\mathbb{F}| = \mathsf{poly}(n)$, the $\mathsf{GapMWSP}_\beta$ problem is NP-hard. Specifically, there exists a deterministic Karp-Levin reduction from SAT to $\mathsf{GapMWSP}_\beta$, where the reduction algorithm takes a target field $\mathbb{F}$ as an explicit input and outputs an instance $(\mathbf{A}, \mathbf{b}, d)$ over $\mathbb{F}$ in time $\mathsf{poly}(n, \log|\mathbb{F}|)$.*

Lemma 4.2 follows by a direct adaptation of [HKLT19, Theorem 2.1], generalized to arbitrary finite fields. We give the proof in Appendix D.

A linear PCP for GapMWSP and a laconic argument for NP. We now show how to construct a simple 1-query (instance-dependent) linear PCP for the GapMWSP problem. Then, together with ElGamal encryption, we can invoke the [BCI+13] compiler (Theorem A.11) to obtain a 2-message laconic argument for GapMWSP, and correspondingly, by Lemma 4.2, for all of NP.

Construction 4.3 (Linear PCP for GapMWSP)**.** Let $\varepsilon > 0$ be a completeness parameter and $\beta > 0$ be the approximation factor. We construct a 1-query *instance-dependent* linear PCP $\Pi_{\mathsf{LPCP}} = (\mathcal{Q}_{\mathsf{LPCP}}, \mathcal{P}_{\mathsf{LPCP}}, \mathcal{D}_{\mathsf{LPCP}})$ for $\mathsf{GapMWSP}_\beta$ as follows:

- $\mathcal{Q}_{\mathsf{LPCP}}(\mathbf{A}, \mathbf{b}, d)$: On input a $\mathsf{GapMWSP}_\beta$ $(\mathbf{A}, \mathbf{b}, d)$ where $\mathbf{A} \in \mathbb{F}^{\ell \times n}$, $\mathbf{b} \in \mathbb{F}^\ell$ and $d \in \mathbb{N}$, the query algorithm does the following:
  1. Sample a random vector $\mathbf{e} \in \mathbb{F}^n$ where each component $e_i = 0$ with probability $1 - \varepsilon/d$ and $e_i \xleftarrow{R} \mathbb{F}$ otherwise.
  2. Sample a random vector $\mathbf{r} \xleftarrow{R} \mathbb{F}^\ell$.

  Output the query $\mathbf{q}^\mathsf{T} = \mathbf{r}^\mathsf{T}\mathbf{A} + \mathbf{e}^\mathsf{T} \in \mathbb{F}^n$ and $\mathsf{st} = \mathbf{r}^\mathsf{T}\mathbf{b} \in \mathbb{F}$.
- $\mathcal{P}_{\mathsf{LPCP}}((\mathbf{A}, \mathbf{b}, d), \mathbf{y})$: On input a $\mathsf{GapMWSP}_\beta$ instance $(\mathbf{A}, \mathbf{b}, d)$ and a witness $\mathbf{y} \in \mathbb{F}^n$, output $\boldsymbol{\pi} = \mathbf{y}$.
- $\mathcal{D}_{\mathsf{LPCP}}(\mathsf{st}, a)$: On input the verification state $\mathsf{st} \in \mathbb{F}$ and a response $a \in \mathbb{F}$, the verifier outputs 1 if $a = \mathsf{st}$ and 0 otherwise.

Theorem 4.4 (Completeness)**.** *Construction 4.3 has completeness error $1/\varepsilon$.*

Proof. Let $(\mathbf{A}, \mathbf{b}, d)$ be a YES instance for $\mathsf{GapMWSP}_\beta$ and $\mathbf{y} \in \mathbb{F}^n$ be a witness. Let $(\mathsf{st}, \mathbf{q}) \leftarrow \mathcal{Q}_{\mathsf{LPCP}}(\mathbf{A}, \mathbf{b}, d)$, $\boldsymbol{\pi} \leftarrow \mathcal{P}_{\mathsf{LPCP}}((\mathbf{A}, \mathbf{b}, d), \mathbf{y})$. By construction, $\boldsymbol{\pi} = \mathbf{y}$ and moreover, $\mathbf{Ay} = \mathbf{b}$ and $\mathsf{wt}(\mathbf{y}) \le d$. This means that

$$\mathbf{q}^\mathsf{T}\boldsymbol{\pi} = \mathbf{r}^\mathsf{T}\mathbf{Ay} + \mathbf{e}^\mathsf{T}\mathbf{y} = \mathbf{r}^\mathsf{T}\mathbf{b} + \mathbf{e}^\mathsf{T}\mathbf{y}.$$

By construction, each component of $\mathbf{e}$ is sampled independently and is zero with probability $1 - \varepsilon/d$. Since $\mathsf{wt}(\mathbf{y}) \le d$, over the randomness of $\mathbf{e}$, we have that

$$\Pr[\mathbf{e}^\mathsf{T}\mathbf{y} = 0] \ge (1 - \varepsilon/d)^{\mathsf{wt}(\mathbf{y})} \ge (1 - \varepsilon/d)^d \ge 1 - \varepsilon.$$

Thus, with probability $1 - \varepsilon$, $\mathbf{e}^\mathsf{T}\mathbf{y} = 0$, and so $\mathbf{q}^\mathsf{T}\boldsymbol{\pi} = \mathbf{r}^\mathsf{T}\mathbf{b}$, and the verifier accepts. □

Theorem 4.5 (Soundness)**.** *If $\varepsilon\beta = \omega(\log n)$, then Construction 4.3 has soundness error $1/|\mathbb{F}| + \mathsf{negl}(n)$.*

Proof. Take any NO instance $(\mathbf{A}, \mathbf{b}, d)$. Consider any proof $\boldsymbol{\pi}^* = \mathbf{y}^* \in \mathbb{F}^n$ and $\delta^* \in \mathbb{F}$. Suppose the verifier accepts. Then, it must be the case that

$$\mathbf{r}^\mathsf{T}\mathbf{b} = \mathbf{q}^\mathsf{T}\boldsymbol{\pi}^* + \delta^* = \mathbf{r}^\mathsf{T}\mathbf{Ay}^* + \mathbf{e}^\mathsf{T}\mathbf{y}^* + \delta^*,$$

or equivalently, that

$$\mathbf{r}^\mathsf{T}(\mathbf{Ay}^* - \mathbf{b}) + \mathbf{e}^\mathsf{T}\mathbf{y}^* + \delta^* = 0. \tag{4.1}$$

We can view Eq. (4.1) as a polynomial in $\mathbf{r}$ and $\mathbf{e}$. We consider two cases:

- Suppose that $\mathbf{A}\mathbf{y}^* \neq \mathbf{b}$. In this case, Eq. (4.1) is a nonzero polynomial in $\mathbf{r}$. Since $\mathbf{r}$ is uniform over $\mathbb{F}^k$ (and independent of all other quantities), by the Schwartz-Zippel lemma,

$$\Pr[\mathbf{r}^\mathsf{T}(\mathbf{A}\mathbf{y}^* - \mathbf{b}) + \mathbf{e}^\mathsf{T}\mathbf{y}^* + \delta^* = 0 \mid \mathbf{r} \xleftarrow{\mathrm{R}} \mathbb{F}^\ell] \leq 1/|\mathbb{F}|.$$

  Thus, in this case, the verifier accepts with probability at most $1/|\mathbb{F}|$.

- Suppose that $\mathbf{A}\mathbf{y}^* = \mathbf{b}$. Since $(\mathbf{A}, \mathbf{b}, d)$ is a NO instance, this means that $\mathsf{wt}(\mathbf{y}^*) \geq \beta \cdot d$. Let $S_{\mathbf{y}^*} = \{i \in [n] : y_i^* \neq 0\}$ be the set of indices in $\mathbf{y}^*$ that are nonzero. By construction, each component e_i for $i \in [n]$ is independent and nonzero with probability ε/d. This means that

$$\Pr[\forall i \in S_{\mathbf{y}^*} : e_i = 0] = (1 - \varepsilon/d)^{\mathsf{wt}(\mathbf{y}^*)} \leq (1 - \varepsilon/d)^{\beta d} \leq e^{-\varepsilon\beta} = \mathsf{negl}(n),$$

  whenever $\varepsilon\beta = \omega(\log n)$. In this case, with probability $1 - \mathsf{negl}(n)$, there exists some $i \in [n]$ where $e_i \neq 0$ and $y_i^* \neq 0$, and correspondingly, the polynomial in Eq. (4.1) is nonzero in e_i. Since e_i is uniform over $\mathbb{F}$ in this case, the probability that Eq. (4.1) holds is $1/|\mathbb{F}|$ by Schwartz-Zippel. Thus, in this case, the verifier accepts with probability at most $1/|\mathbb{F}| + \mathsf{negl}(n)$.

We conclude that over the verifier's randomness, for a false statement $x \notin \mathcal{L}$, the verifier accepts with probability at most $1/|\mathbb{F}| + \mathsf{negl}(n)$. □

Corollary 4.6 (1-Query Linear PCP for NP)**.** *Let $\mathbb{F}$ be a finite field, let $\mathcal{C} = \{C_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of Boolean circuits where each circuit C_λ has size $s = s(\lambda)$. Let $\mathcal{L}_\mathcal{C}$ be the associated language of Boolean circuit satisfiability. There exists an instance-dependent 1-query linear PCP for $\mathcal{L}_\mathcal{C}$ over $\mathbb{F}$ with completeness error $o(1)$, and soundness error $1/|\mathbb{F}| + \mathsf{negl}(\lambda)$. The query size is $\mathsf{poly}(\lambda, s)$.*

Proof. Take any constant $c > 2$. We instantiate Construction 4.3 with $\beta(\lambda) = \log^c \lambda$ and $\varepsilon(\lambda) = 1/\sqrt{\beta(\lambda)} = o(1)$. In this case, $\varepsilon\beta = \omega(\log \lambda)$, so by Theorems 4.4 and 4.5, Construction 4.3 is a linear PCP for $\mathsf{GapMWSP}_\beta$ with completeness error $o(1)$ and soundness error $1/|\mathbb{F}| + \mathsf{negl}(\lambda)$. The claim now follows by Lemma 4.2. □

Corollary 4.7 (Laconic Argument for NP from ElGamal)**.** *If the ElGamal encryption scheme over a group $\mathbb{G}$ (Construction C.4) satisfies linear targeted malleability with respect to singleton subsets (Definition A.6), there exists a 2-message laconic argument for NP with completeness error $o(1)$ and negligible soundness error where the prover's message consists of 2 group elements (i.e., has size $2\log|\mathbb{G}|$).*

Proof. The linear PCP from Construction 4.3 and Corollary 4.6 has the property that for every query, there is a *single* response that the verifier accepts, and moreover, this is known at query-generation time. Thus, if the ElGamal encryption scheme satisfies linear targeted malleability with respect to sets of size 1, then the claim follows from Theorem A.11 and Remark A.12. □

Remark 4.8 (Algebraic Degree of the Verifier in Construction 4.3)**.** The linear PCP for GapMWSP from Construction 4.3 has the property that the algebraic degree of the verifier's decision algorithm is *linear*. This means we can potentially apply the pairing-based compiler from [BCI+13] to obtain a laconic argument for GapMWSP (and NP) with a "predictable" prover message (i.e., there is a single prover message that the verifier accepts). Such argument systems directly imply witness encryption for the underlying language [FNV17] (see Section 5 for a more comprehensive discussion). Unfortunately, the pairing-based compiler from [BCI+13] cannot be applied to Construction 4.3 because the algebraic degree of the query-generation algorithm in the linear PCP is *super-polynomial*. Specifically, the components of the error vector $\mathbf{e} \in \mathbb{F}^n$ in Construction 4.3 is zero with noticeable probability. Thus, the minimal degree of any polynomial that computes the components of $\mathbf{e}$ is at least $|\mathbb{F}|/\mathsf{poly}(\lambda)$, which is super-polynomial when $|\mathbb{F}|$ is super-polynomial. The pairing-based compiler in [BCI+13] only applies to linear PCPs where the algebraic degree of the query-generation algorithm is $\mathsf{poly}(\lambda)$ (and the algebraic degree of the decision algorithm is at most 2). In contrast, the compilers in the designated-verifier setting based on linear-only encryption do not impose any restrictions on the algebraic degree of the query-generation or decision algorithms.

Remark 4.9 (On the Lower Bound from [Gro16])**.** Previously, Groth [Gro16] gave a lower bound against the existence of any 2-message linear interactive proof with a linear decision procedure for any hard language $\mathcal{L}$. Construction 4.3 gives a 1-query linear PCP for general NP with a linear decision procedure. Our construction is *not* in conflict with the lower bound of [Gro16] because it does not satisfy *perfect completeness*, while the lower bound of [Gro16] relied on the underlying proof having sufficiently small completeness error.

Here, we provide some high-level intuition for why this is the case. For simplicity, we present the lower-bound argument from [Gro16] for the special case of 1-query linear PCPs. Take a hard NP language $\mathcal{L}$ (i.e., one where YES instances are hard to distinguish from NO instances), and consider a 1-query linear PCP $(\mathcal{Q}_{\mathsf{LPCP}}, \mathcal{P}_{\mathsf{LPCP}}, \mathcal{D}_{\mathsf{LPCP}})$ for $\mathcal{L}$ over a finite field $\mathbb{F}$. Suppose $(\mathsf{st}, \mathbf{q}) \leftarrow \mathcal{Q}_{\mathsf{LPCP}}$. For a proof $\boldsymbol{\pi} \in \mathbb{F}^\ell$, the verifier's decision algorithm takes as input the response $a = \mathbf{q}^\mathsf{T}\boldsymbol{\pi} \in \mathbb{F}$ together with the statement $\mathbf{x}$ and the verification state st. If the verifier's decision procedure is linear, then the verification relation corresponds to testing $a \stackrel{?}{=} b$ where $b \in \mathbb{F}$ is a function of the state st and the statement $\mathbf{x}$ (but independent of the proof $\boldsymbol{\pi}$). Correspondingly, this means that the proof $\boldsymbol{\pi}$ must satisfy the linear relation $\mathbf{q}^\mathsf{T}\boldsymbol{\pi} = b$.

This yields the following algorithm for deciding membership of $\mathbf{x}$. Let $m = \ell \cdot \lceil \log|\mathbb{F}| \rceil$. For $i \in [m]$, sample $(\mathsf{st}_i, \mathbf{q}_i) \leftarrow \mathcal{Q}_{\mathsf{LPCP}}$, and compute the verification target $b_i \in \mathbb{F}$ from $\mathbf{x}$ and st_i. Let $\mathbf{Q} \in \mathbb{F}^{m \times \ell}$ be the matrix whose rows are the vectors $\mathbf{q}_1^\mathsf{T}, \ldots, \mathbf{q}_m^\mathsf{T}$ and let $\mathbf{b} \in \mathbb{F}^m$ be the vector whose entries are $b_1, \ldots, b_m$. The decision algorithm outputs 1 if there exists $\boldsymbol{\pi} \in \mathbb{F}^\ell$ such that $\mathbf{Q}\boldsymbol{\pi} = \mathbf{b}$ and 0 otherwise. Now, if the completeness error of the underlying linear PCP is negligible (or zero), then for a statement $\mathbf{x} \in \mathcal{L}$, such a proof $\boldsymbol{\pi}$ exists with overwhelming probability (namely, the honestly-generated proof will pass all of the verification relations). Conversely, when $\mathbf{x} \notin \mathcal{L}$, soundness of the linear PCP ensures that any fixed string $\boldsymbol{\pi} \in \mathbb{F}^\ell$ will satisfy the i^{th} verification procedure with negligible probability. Applying a union bound over all possible proof strings $\boldsymbol{\pi} \in \mathbb{F}^\ell$, the probability that there exists $\boldsymbol{\pi} \in \mathbb{F}_p^\ell$ where $\mathbf{Q}\boldsymbol{\pi} = \mathbf{b}$ is negligible.

The above algorithm distinguishes between YES instances and NO instances by checking whether there is a solution $\boldsymbol{\pi} \in \mathbb{F}^\ell$ that satisfies the linear system $\mathbf{Q}\boldsymbol{\pi} = \mathbf{b}$. However, when the proof system has *noticeable* completeness error, the proof $\boldsymbol{\pi}$ output by an honest prover may fail to satisfy the verification relation $\mathbf{q}_i^\mathsf{T}\boldsymbol{\pi} = b_i$ with noticeable probability. In this case, there is no longer a guarantee that for YES instances, a solution $\boldsymbol{\pi} \in \mathbb{F}^\ell$ to the linear system exists, and as such, the above distinguisher no longer applies.

5 1-Element Laconic Arguments and Witness Encryption

In this section, we first show that any laconic argument system for an NP language $\mathcal{L}$ (with negligible soundness error) where the proof consists of a single group element (i.e., a "1-element laconic argument") and where the verification algorithm can be modeled as a "generic" algorithm implies a witness encryption scheme for $\mathcal{L}$. Note that since the prover is restricted to sending a single group element, this effectively restricts the prover to sending at most one message in the protocol. Thus, it suffices to just consider *2-message* laconic arguments here. Our construction of witness encryption proceeds in two steps. We first show that any laconic argument satisfying the above properties must be *predictable* [FNV17]. We then invoke the Faonio et al. [FNV17] compiler on the predictable argument to obtain a witness encryption scheme.

Next, we show that under a new hypothesis on the hardness of approximating the minimum distance of a linear code [DMS99] (Hypothesis 5.12), we can construct a laconic element for NP where the proof consists of a single group element in the generic group model. Thus, under our hypothesis, we obtain a witness encryption scheme for NP in the generic group model.

5.1 Predictable Arguments and Witness Encryption

We begin by reviewing the notion of a witness encryption scheme [GGSW13] as well as the notion of a predictable argument [FNV17] and their connections to witness encryption.

Definition 5.1 (Witness Encryption [GGSW13])**.** A witness encryption for an NP relation $\mathcal{R}$ (and associated language $\mathcal{L}$) and correctness error $c = c(\lambda)$ is a tuple $\Pi_{\mathsf{WE}} = (\mathsf{Encrypt}, \mathsf{Decrypt})$ with the following syntax:

- $\mathsf{Encrypt}(1^\lambda, x, m) \to \mathsf{ct}$: On input the security parameter λ, a statement x, and a message $m \in \{0,1\}$, the encryption algorithm outputs a ciphertext ct.
- $\mathsf{Decrypt}(w, \mathsf{ct}) \to m$: On input a witness w and a ciphertext ct, the decryption algorithm outputs a message $m \in \{0,1\} \cup \{\bot\}$.

Moreover, Π_{WE} should satisfy the following properties:

- **Correctness:** For every $\lambda \in \mathbb{N}$, every message $m \in \{0,1\}$, and every $(x, w) \in \mathcal{R}$,

$$\Pr[\mathsf{Decrypt}(w, \mathsf{Encrypt}(1^\lambda, x, m)) = m] \geq 1 - c(\lambda).$$

  We say that Π_{WE} satisfies statistical correctness if $c(\lambda) = \mathsf{negl}(\lambda)$ and perfect correctness if $c(\lambda) = 0$.
- **Semantic security:** For every efficient adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $x \notin \mathcal{L}$,

$$\Pr[\mathcal{A}(1^\lambda, \mathsf{ct}_b) = b \mid b \xleftarrow{R} \{0,1\}, \mathsf{ct}_b \leftarrow \mathsf{Encrypt}(1^\lambda, x, b)] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

Remark 5.2 (Correcting Decryption Errors)**.** Any witness encryption scheme $\Pi_{\mathsf{WE}} = (\mathsf{Encrypt}, \mathsf{Decrypt})$ with correctness error $c(\lambda) < 1/2 - \delta$ where $\delta = 1/\mathsf{poly}(\lambda)$ can be boosted into a witness encryption scheme with negligible correctness error. Namely, we take the encryption of a message m (with statement x) to be $t = O(\lambda/\delta^2)$ independent encryptions of m (with statement x) under Π_{WE}. The decryption algorithm would then decrypt each of the t ciphertexts and output the majority of the t decrypted bits. Statistical correctness now follows by Hoeffding's inequality.

Predictable arguments. Faonio et al. [FNV17] previously showed that a predictable argument of knowledge for an NP language $\mathcal{L}$ implies an extractable witness encryption scheme for $\mathcal{L}$. In the same work, they also note that the same construction can be used to obtain a standard (non-extractable) witness encryption scheme for $\mathcal{L}$ from any predictable argument for $\mathcal{L}$. Here, a predictable argument is one where on each round, there is a single message that causes the verifier to accept, and moreover, this message is known to the verifier *a priori*. In our setting, we focus on argument systems where the prover can only send a single group element, so it suffices to just consider 2-message arguments. While Faonio et al. focused on the setting where the predictable argument satisfies perfect completeness (and negligible soundness), the same transformation still applies even if the scheme has a large completeness error (specifically, it suffices that $1 - c = 1/\mathsf{poly}(\lambda)$, where c is the completeness error). We provide the definitions and main theorem statements below.

Definition 5.3 (Predictable Laconic Argument [FNV17, adapted])**.** A 2-message laconic argument $\Pi_{\mathsf{LA}} = (\mathcal{Q}_{\mathsf{LA}}, \mathcal{P}_{\mathsf{LA}}, \mathcal{V}_{\mathsf{LA}})$ is *predictable* if $\mathcal{V}_{\mathsf{LA}}(\mathsf{st}, \pi)$ outputs 1 if and only if $\mathsf{st} = \pi$. Namely, on input the security parameter λ and an instance x, the query algorithm $\mathcal{Q}_{\mathsf{LA}}$ samples a query q together with a target proof string π, which is set as the verifier's state.

Theorem 5.4 (Predictable Arguments to Witness Encryption [FNV17, adapted])**.** *Let $\Pi_{\mathsf{LA}} = (\mathcal{Q}_{\mathsf{LA}}, \mathcal{P}_{\mathsf{LA}}, \mathcal{V}_{\mathsf{LA}})$ be a 2-message predictable argument for an NP language $\mathcal{L}$ with completeness error c and soundness error $\varepsilon = \mathsf{negl}(\lambda)$. Then, there exists a witness encryption scheme for $\mathcal{L}$ with correctness error $c/2$.*

We give a proof of Theorem 5.4 in Appendix E. The proof follows the structure of the corresponding proof in [FNV17], just extended to the setting of imperfect completeness.

Arguments with generic verification. In the following, we will focus exclusively on arguments with a "generic" verification procedure. We say that a 2-message argument has a generic verification algorithm if the prover's message consists of a tuple of group elements, and the verifier's decision procedure only operates on the group elements in a generic manner (i.e., the verifier can only evaluate the group operation and test if two group elements are equal). We also allow the verifier to compute an *arbitrary* function (modeled as a Boolean circuit) of the results of the equality checks it performs on the group elements. For example, this model captures the verification algorithm of our ElGamal-based SNARGs from Section 3.2 (and more generally, *any* instantiation of the [BCI+13] compiler using ElGamal encryption). In fact, *all* of the group-based arguments we develop in this work have a generic verification procedure under our definition.

A similar notion of argument systems with a generic verification procedure was considered in [Gro16] in the pairing-based setting. The [Gro16] model was more restrictive in that it required that the verifier accept only if *all* of the (pairing-based) relations in the decision algorithm were satisfied. In our setting, we allow the verifier's decision to be any efficiently-computable function of the equality checks.

Definition 5.5 (Laconic Arguments with Generic Verification)**.** Let $\Pi_{\mathsf{LA}} = (\mathcal{Q}_{\mathsf{LA}}, \mathcal{P}_{\mathsf{LA}}, \mathcal{V}_{\mathsf{LA}})$ be a 2-message laconic argument over a group $\mathbb{G}$ of prime-order p with generator g. We say that Π_{LA} has a generic verification algorithm if the following properties hold:

- The state st output by $\mathcal{Q}_{\mathsf{LA}}$ consists of the description of the group $(\mathbb{G}, p, g)$, a collection of nonzero vectors $\mathbf{0} \neq \mathbf{a}_i \in \mathbb{F}_p^k$, scalars $b_i \in \mathbb{F}_p$ for $i \in [m]$, and the description of a Boolean circuit $C\colon \{0,1\}^m \to \{0,1\}$. Note that the components $\{\mathbf{a}_i, b_i\}_{i\in[m]}$ as well as the circuit C can depend on the statement x.
- The proof output by $\mathcal{P}_{\mathsf{LA}}$ is a vector of group elements $g^{\boldsymbol{\pi}} \in \mathbb{G}^k$ where $g^{\boldsymbol{\pi}} = (g^{\pi_1}, \ldots, g^{\pi_k})$.
- The verification algorithm $\mathcal{V}_{\mathsf{LA}}$ proceeds in two stages. On input a proof $g^{\boldsymbol{\pi}} \in \mathbb{G}^k$ and the verification state $\mathsf{st} = (\{\mathbf{a}_i, b_i\}_{i\in[m]}, C)$, the verification algorithm sets $\beta_i = 1$ if $g^{\mathbf{a}_i^\intercal \boldsymbol{\pi}} = g^{b_i}$ and $\beta_i = 0$ otherwise. Then, the verification algorithm computes and outputs $C(\beta_1, \ldots, \beta_m)$.

Weak predictability of 1-element laconic arguments. We now show that argument systems with a generic verification procedure and where the proof consists of a single group element must be predictable. We do this in two steps. First, we show that any 1-element laconic argument satisfies a notion of "weak predictability," where we essentially allow there to be up to k accepting proofs (Definition 5.6). Second, we show that any laconic argument that is weakly predictable is also predictable, but with a larger completeness error (Theorem 5.9).

Definition 5.6 (Weakly-Predictable Laconic Argument)**.** Let $\Pi_{\mathsf{LA}} = (\mathcal{Q}_{\mathsf{LA}}, \mathcal{P}_{\mathsf{LA}}, \mathcal{V}_{\mathsf{LA}})$ be a 2-message laconic argument for a relation $\mathcal{R}$, and let T be the codomain of $\mathcal{P}_{\mathsf{LA}}$ (i.e., the set of possible proof strings). We say that Π_{LA} is *k-weakly predictable* if there exists an efficient predictor algorithm $\mathsf{LA}_{\mathsf{predict}}$ that on input any verification state st in the support of $\mathcal{Q}_{\mathsf{LA}}$, outputs a (possibly empty) set of proofs $S \subseteq T$ with the following two properties:

- Either $|S| \leq k$ or $S \geq |T| - k$.
- $\mathcal{V}_{\mathsf{LA}}(\mathsf{st}, \pi) = 1$ if and only if $\pi \in S$.

Theorem 5.7 (Weak Predictability of 1-Element Laconic Arguments)**.** *Every 1-element laconic argument for a relation $\mathcal{R}$ (that is computable by a Boolean circuit of size $\mathsf{poly}(\lambda)$) with a generic verification algorithm (Definition 5.5) is k-weakly predictable for $k = \mathsf{poly}(\lambda)$.*

Proof. Let $\Pi_{\mathsf{LA}} = (\mathcal{Q}_{\mathsf{LA}}, \mathcal{P}_{\mathsf{LA}}, \mathcal{V}_{\mathsf{LA}})$ be a 1-element laconic argument with a generic verification procedure. We construct a predictor algorithm $\mathsf{LA}_{\mathsf{predict}}$ for Π_{LA} that works as follows:

1. On input $\mathsf{st} = ((\mathbb{G}, p, g), \{a_i, b_i\}_{i\in[m]}, C)$ where $0 \neq a_i \in \mathbb{F}_p$ and $b_i \in \mathbb{F}_p$, compute $\pi_i \leftarrow a_i^{-1} b_i \in \mathbb{F}_p$ for each $i \in [m]$. Note that the $a_1, \ldots, a_m$ are *scalars* in this case because the proof consists of a *single* group element.

2. Initialize a set $T \leftarrow \varnothing$. For each $i \in [m]$, add $g^{\pi_i} \in \mathbb{G}$ to T if $\mathcal{V}_{\mathsf{LA}}(\mathsf{st}, g^{\pi_i}) = 1$.

3. Finally, take any $\pi' \in \mathbb{F}_p \setminus \{\pi_1, \ldots, \pi_m\}$. If no such π' exists or $\mathcal{V}_{\mathsf{LA}}(\mathsf{st}, g^{\pi'}) = 0$, output $S = T$. Otherwise, output the set $S = T \cup (\mathbb{G} \setminus \{g^{\pi_1}, \ldots, g^{\pi_m}\})$.

It suffices to argue that $\mathsf{LA}_{\mathsf{predict}}$ satisfies each of the required properties. Since $\mathcal{Q}_{\mathsf{LA}}$ is efficient, we have that $m = \mathsf{poly}(\lambda)$, and so $\mathsf{LA}_{\mathsf{predict}}$ is efficient. By construction, the output S of $\mathsf{LA}_{\mathsf{predict}}$ satisfies $|S| \leq m$ or $|S| \geq |\mathbb{G}| - m$, so the first property is satisfied. For the second property, take any candidate proof $g^{\pi} \in \mathbb{G}$ (where $\pi \in \mathbb{F}_p$). We consider two cases:

- Suppose $\pi \in \{\pi_1, \ldots, \pi_m\}$. Then, $g^{\pi} \in T$ if and only if $\mathcal{V}_{\mathsf{LA}}(\mathsf{st}, g^{\pi}) = 1$. Since $T \subseteq S$, the claim holds.
- Suppose $\pi \notin \{\pi_1, \ldots, \pi_m\}$. This means that $a_i \pi \neq b_i \in \mathbb{F}_p$ for all $i \in [m]$. By definition then, $\mathcal{V}_{\mathsf{LA}}(\mathsf{st}, g^{\pi}) = C(0^m) = \mathcal{V}_{\mathsf{LA}}(\mathsf{st}, g^{\pi'})$. Correspondingly, $g^{\pi} \in S$ if and only if $\mathcal{V}_{\mathsf{LA}}(\mathsf{st}, g^{\pi'}) = 1 = \mathcal{V}_{\mathsf{LA}}(\mathsf{st}, g^{\pi})$, and again, the claim holds. □

From weak predictability to predictability. Now we show that any laconic argument that is weakly predictable is predictable (Definition 5.3), though with a much larger completeness error. The construction is simple: to obtain a standard predictable argument from a weakly-predictable one, the query-generation algorithm samples one of the k possible proofs in the weakly-predictable scheme at random as its *only* accepting proof. While this transformation substantially increases the completeness error, it still suffices to obtain witness encryption (as long as $k = \mathsf{poly}(\lambda)$). Combining these results with Theorem 5.4, we show that any 1-element laconic argument for an NP language $\mathcal{L}$ where verification is generic and which has negligible soundness error implies a witness encryption for $\mathcal{L}$ (Corollary 5.10).

Construction 5.8 (Predictable Argument from Weakly-Predictable Argument)**.** Let $\Pi'_{\mathsf{LA}} = (\mathcal{Q}'_{\mathsf{LA}}, \mathcal{P}'_{\mathsf{LA}}, \mathcal{V}'_{\mathsf{LA}})$ be a k-weakly predictable 2-message laconic argument for a relation $\mathcal{R}$, and let $\mathsf{LA}'_{\mathsf{predict}}$ be the predictor algorithm for Π'_{LA}. We construct a predictable laconic argument $\Pi_{\mathsf{LA}} = (\mathcal{Q}_{\mathsf{LA}}, \mathcal{P}_{\mathsf{LA}}, \mathcal{V}_{\mathsf{LA}})$ for $\mathcal{R}$ as follows:

- $\mathcal{Q}_{\mathsf{LA}}(1^\lambda, x)$: On input the security parameter λ and a statement x, the query-generation algorithm computes $(\mathsf{st}', q') \leftarrow \mathcal{Q}'_{\mathsf{LA}}(1^\lambda, x)$ and runs $S \leftarrow \mathsf{LA}'_{\mathsf{predict}}(\mathsf{st}')$. If $1 \leq |S| \leq k$, it samples $\pi \xleftarrow{\mathrm{R}} S$ and outputs the query $q = q'$ and the state $\mathsf{st} = \pi$. If $|S| = 0$, it samples $\pi \xleftarrow{\mathrm{R}} T$ and outputs the query $q = q'$ and the state $\mathsf{st} = \pi$. Finally, if $|S| > k$, it output the query $q = \bot$ and the state $\mathsf{st} = \bot$.
- $\mathcal{P}_{\mathsf{LA}}(q, x, w)$: On input a query q, the statement x, and a witness w, the prover algorithm outputs the proof $\pi = \bot$ if $q = \bot$. Otherwise, it computes and outputs $\pi \leftarrow \mathcal{P}'_{\mathsf{LA}}(q, x, w)$.
- $\mathcal{V}_{\mathsf{LA}}(\mathsf{st}, \pi)$: On input the verification state st and a proof π, the verification algorithm outputs 1 if $\pi = \mathsf{st}$ and 0 otherwise.

Theorem 5.9 (Predictable Argument from Weakly-Predictable Argument)**.** *Let Π'_{LA} be a k-weakly predictable argument for a relation $\mathcal{R}$ (with associated language $\mathcal{L}$) with completeness error c' and soundness error ε'. Let T be the codomain of $\mathcal{P}'_{\mathsf{LA}}$ and suppose that $|T| \geq 2 \cdot k$. Then the laconic argument Π_{LA} from Construction 5.8 is a predictable argument for $\mathcal{R}$ with completeness error $c = 1 - (1 - c')/k$ and soundness error $\varepsilon = 3 \cdot \varepsilon' + 1/|T|$.*

Proof. By construction, Π_{LA} is predictable, so it suffices to argue completeness and soundness:

- **Completeness**: Take any $(x, w) \in \mathcal{R}$. By definition, the query-generation algorithm $\mathcal{Q}_{\mathsf{LA}}(1^\lambda, x)$ starts by computing $S \leftarrow \mathsf{LA}'_{\mathsf{predict}}(\mathsf{st}')$ where $(\mathsf{st}', q') \leftarrow \mathcal{Q}'_{\mathsf{LA}}(1^\lambda, x)$. Since Π'_{LA} is k-weakly predictable, this means that either $|S| \leq k$ or $|S| > k$. We define the following probabilities:

$$\delta_1 \coloneqq \Pr[|S| \leq k \mid (\mathsf{st}', q') \leftarrow \mathcal{Q}'_{\mathsf{LA}}(1^\lambda, x); S \leftarrow \mathsf{LA}'_{\mathsf{predict}}(\mathsf{st}')]$$

$$\delta_2 \coloneqq \Pr[|S| > k \mid (\mathsf{st}', q') \leftarrow \mathcal{Q}'_{\mathsf{LA}}(1^\lambda, x); S \leftarrow \mathsf{LA}'_{\mathsf{predict}}(\mathsf{st}')]$$

$$p_1 \coloneqq \Pr[\mathcal{V}'_{\mathsf{LA}}(\mathsf{st}', x, \pi) = 1 \wedge |S| \leq k \mid (\mathsf{st}', q') \leftarrow \mathcal{Q}'_{\mathsf{LA}}(1^\lambda, x); S \leftarrow \mathsf{LA}'_{\mathsf{predict}}(\mathsf{st}'); \pi \leftarrow \mathcal{P}'_{\mathsf{LA}}(x, q')]$$

$$p_2 \coloneqq \Pr[\mathcal{V}'_{\mathsf{LA}}(\mathsf{st}', x, \pi) = 1 \wedge |S| > k \mid (\mathsf{st}', q') \leftarrow \mathcal{Q}'_{\mathsf{LA}}(1^\lambda, x); S \leftarrow \mathsf{LA}'_{\mathsf{predict}}(\mathsf{st}'); \pi \leftarrow \mathcal{P}'_{\mathsf{LA}}(x, q')]$$

Note that if $|S| = 0$, then $\mathcal{V}'_{\mathsf{LA}}$ never accepts, so by definition, $p_1 + p_2 \geq 1 - c'$. It suffices to consider the following two cases:

- If $1 \leq |S| \leq k$, then $q = q'$ and the prover algorithm computes the proof as $\pi \leftarrow \mathcal{P}'_{\mathsf{LA}}(q, x, w)$. This means that $\mathcal{V}'_{\mathsf{LA}}(\mathsf{st}', x, \pi) = 1$ with probability p_1/δ_1. By definition of S, this means that $\pi \in S$ with probability at least p_1/δ_1. Since st is sampled uniformly at random from S and $|S| \leq k$, we have that
$$\Pr[\mathcal{V}_{\mathsf{LA}}(\mathsf{st}, \pi) = 1] = \Pr[\pi = \mathsf{st}] \geq p_1/(\delta k).$$
- If $|S| > k$, then $q = \bot$ and the prover algorithm computes the proof as $\pi = \bot$. In this case, the verifier accepts with probability 1.

The verifier thus accepts with probability at least $p_1/(\delta_1 k) \cdot \delta_1 + \delta_2 \geq p_1/k + p_2 \geq (p_1 + p_2)/k \geq (1 - c')/k$. The completeness error is then $1 - (1 - c')/k$.

- **Soundness:** Let $\mathcal{P}^*$ be any polynomial-size prover for Π_{LA}. Take any $x \notin \mathcal{L}$, and consider $(q, \mathsf{st}) \leftarrow \mathcal{Q}_{\mathsf{LA}}(1^\lambda, x)$. By definition, $\mathcal{Q}_{\mathsf{LA}}$ first computes $(q', \mathsf{st}') \leftarrow \mathcal{Q}'_{\mathsf{LA}}(1^\lambda, x)$ and $S \leftarrow \mathsf{LA}'_{\mathsf{predict}}(\mathsf{st}')$. We consider two possibilities:
  - If $|S| = 0$ then st is a random element from T and independent of the query $q = q'$. Thus, the probability that the proof π output by the prover satisfies $\pi = \mathsf{st}$ is at most $1/|T|$.
  - If $1 \leq |S| \leq k$, then $q = q'$. By soundness of Π'_{LA}
$$\Pr[\mathcal{V}'_{\mathsf{LA}}(\mathsf{st}', \pi) = 1 \mid (q', \mathsf{st}') \leftarrow \mathcal{Q}'_{\mathsf{LA}}(1^\lambda, x), \pi \leftarrow \mathcal{P}^*(1^\lambda, x, q')] \leq \varepsilon'.$$
    By construction of $\mathsf{LA}'_{\mathsf{predict}}$, we have that $\pi \in S$ if and only if $\mathcal{V}'_{\mathsf{LA}}(\mathsf{st}', \pi) = 1$. Thus, with probability at most ε', the proof π output by $\mathcal{P}^*$ in this case is contained in S. Correspondingly, the probability that $\pi = \mathsf{st} \in S$ is at most ε'.
  - Suppose $|S| > k$. By definition of weak predictability, this means that $|S| \geq |T| - k$. We argue that this case occurs with probability at most $2\varepsilon'$ (over the randomness of query-generation). Suppose otherwise, and consider a prover for Π'_{LA} that simply outputs a random proof string $\pi \xleftarrow{R} T$. If $|S| \geq |T| - k$, then with probability $|S|/|T| \geq (|T| - k)/|T| \geq 1/2$, we have that $\pi \in S$, and $\mathcal{V}'_{\mathsf{LA}}(\mathsf{st}', \pi)$ outputs 1. Thus, if $\mathcal{Q}'_{\mathsf{LA}}(1^\lambda, x)$ outputs st' such that $S \leftarrow \mathsf{LA}'_{\mathsf{predict}}(\mathsf{st}')$ satisfies $|S| \geq k$ with probability at least $2\varepsilon'$, then this prover breaks soundness of Π'_{LA} with probability at least ε', which is a contradiction. Hence, this case can only happen with probability at most $2\varepsilon'$.

  We conclude that if Π'_{LA} has soundness error ε', then Π_{LA} has soundness error at most $3\varepsilon' + 1/|T|$. □

Corollary 5.10 (Witness Encryption from 1-Element Laconic Argument)**.** *Let $\Pi_{\mathsf{LA}} = (\mathcal{Q}_{\mathsf{LA}}, \mathcal{P}_{\mathsf{LA}}, \mathcal{V}_{\mathsf{LA}})$ be a laconic argument for an NP language $\mathcal{L}$ where the proof consists of a single group element and the verification algorithm is generic (Definition 5.5). If Π_{LA} has completeness error $c = c(\lambda)$ where $1 - c \geq 1/\mathsf{poly}(\lambda)$ and negligible soundness error $\varepsilon(\lambda) = \mathsf{negl}(\lambda)$, then there exists a witness encryption scheme for $\mathcal{L}$ with statistical correctness.*

Proof. This follows by combining Theorems 5.4, 5.7 and 5.9. Namely, by Theorem 5.7, Π_{LA} is k-weakly predictable for some $k = \mathsf{poly}(\lambda)$. From Construction 5.8 and Theorem 5.9, we obtain a predictable laconic argument Π'_{LA} for $\mathcal{L}$ with completeness error $c' = 1 - (1 - c)/k$ and soundness $\varepsilon' = 3\varepsilon + 1/|\mathbb{G}|$, where $\mathbb{G}$ is the group associated with Π_{LA}. Since ε is negligible, so is ε' (since $1/|\mathbb{G}|$ is negligible). Thus, we can apply Theorem 5.4 to Π'_{LA} to obtain a witness encryption scheme for $\mathcal{L}$ with correctness error $c'/2 = 1/2 - (1 - c)/(2k)$. Since $(1 - c) \geq 1/\mathsf{poly}(\lambda)$ and $k = \mathsf{poly}(\lambda)$, this means that $c'/2 = 1/2 - 1/\mathsf{poly}(\lambda)$, so we can use parallel repetition to amplify correctness (Remark 5.2). □

5.2 1-Element Laconic Argument from Hardness of Approximation

In this section, we show that under a hardness of approximation hypothesis for the minimum distance problem, which strengthens known hardness results [DMS99], we can construct a 1-element laconic argument for general NP languages. We begin by recalling the minimum distance problem and then stating our hardness of approximation hypothesis.

Definition 5.11 (Gap Minimum Distance Problem (GapMDP) [DMS99, Definition 1])**.** For an approximation factor β, an instance of the GapMDP_β problem is a pair $(\mathbf{A}, d)$ where $\mathbf{A} \in \mathbb{F}^{n\times k}$ for some finite field $\mathbb{F}$ and $d \in \mathbb{N}$ such that

- $(\mathbf{A}, d)$ is a YES instance if $\mathsf{dist}(\mathbf{A}) \le d$.
- $(\mathbf{A}, d)$ is a NO instance if $\mathsf{dist}(\mathbf{A}) \ge \beta \cdot d$.

Here, $\mathsf{dist}(\mathbf{A})$ is the minimum distance (under the Hamming metric) of the code generated by $\mathbf{A}$. We define the GapMDP_β relation to be the NP relation associated with this promise problem (see Section 2 for a formal definition) that takes as input an instance $(\mathbf{A}, d)$ where $\mathbf{A} \in \mathbb{F}^{n\times k}$, and a witness $\mathbf{v} \in \mathbb{F}^k$, and outputs 1 if the following two properties hold:

- $0 < \mathsf{wt}(\mathbf{v}) \le d$;
- $\mathbf{v} \in \mathbb{F}^k$ is a codeword in the code generated by $\mathbf{A}$.

Hypothesis 5.12 (Hardness of Approximation for GapMDP)**.** For some $\beta = \omega(\log n)$, the GapMDP_β relation is NP-hard for any choice of finite field $\mathbb{F}$ where $|\mathbb{F}| = 2^{O(n)}$. Specifically, there exists a deterministic Karp-Levin reduction from SAT to the NP relation associated with the GapMDP_β promise problem,[14] where the reduction algorithm takes a target field $\mathbb{F}$ as an *explicit* input and outputs an instance $(\mathbf{A}, d)$ over $\mathbb{F}$ in time $\mathsf{poly}(n, \log|\mathbb{F}|)$.

Existing hardness results on GapMDP. Dumer et al. [DMS99] showed that the GapMDP was NP-hard for any constant approximation factor $\beta = O(1)$ (over any polynomial-size field) via a randomized reduction. Subsequently, Cheng and Wan [CW09] as well as Austrin and Khot [AK14] gave a deterministic reduction for the same parameter regimes. The latter results additionally give a deterministic *quasi-polynomial* time reduction from NP to the GapMDP_β for any $\beta = 2^{\log^{1-\varepsilon}(n)}$ (i.e., unless $\mathsf{NP} \subseteq \mathsf{DTIME}(2^{\mathsf{polylog}(n)})$, there is no polynomial-time algorithm for GapMDP_β). In our setting, we need to strengthen the existing hardness of approximation results in two different directions: (1) Hypothesis 5.12 requires a deterministic *polynomial* time reduction to GapMDP while the existing reductions in the super-constant regime are all quasi-polynomial; and (2) we require that the reduction applies to large prime characteristic fields (i.e., fields that are super-polynomial in the instance size). While existing reductions are agnostic about the choice of the field, the running time of existing reductions scale polynomially in the characteristic of the field, so they do not directly generalize to super-polynomial size fields.[15] While existing techniques do not suffice for proving Hypothesis 5.12, there are no known barriers to doing so [Kho20]. Finally, we note that existing reductions from NP to GapMDP_β are Karp-Levin reductions; namely, existing reductions show how to map NP instances to GapMDP_β instances (that respect the promise), and moreover, construct an explicit witness for GapMDP_β from a witness for the NP-hard problem.

Construction 5.13 (Laconic Argument for GapMDP)**.** Let λ be a security parameter, $\varepsilon > 0$ be a completeness parameter, and $\beta > 0$ be the approximation factor. Let $\mathsf{GroupGen}$ be a prime-order group generator, and let $p = p(\lambda)$ be the order of the group output by $\mathsf{GroupGen}$. We construct a two-message laconic argument $\Pi_{\mathsf{LA}} = (\mathcal{Q}_{\mathsf{LA}}, \mathcal{P}_{\mathsf{LA}}, \mathcal{V}_{\mathsf{LA}})$ for GapMDP_β (for instances over $\mathbb{F}_p$):

[14] We refer to Section 2 for a formal definition of Karp-Levin reductions from an NP problem to an NP promise problem.
[15] We formulate the hypothesis for $|\mathbb{F}| = 2^{O(n)}$, although any field of super-polynomial size would also suffice.

- $\mathcal{Q}_{\mathsf{LA}}(1^\lambda, (\mathbf{A}, d))$: On input the security parameter λ and an GapMDP_β instance $(\mathbf{A}, d)$ over $\mathbb{F}_p$, the query algorithm samples $(\mathbb{G}, p, g) \leftarrow \mathsf{GroupGen}(1^\lambda)$. Let $\mathbf{H} \in \mathbb{F}_p^{\ell \times k}$ be the parity check matrix for the code generated by $\mathbf{A}$. Then the verifier constructs the following components:
  - Sample a random vector $\mathbf{e} \in \mathbb{F}_p^k$ where each component $e_i = 0$ with probability $1 - \varepsilon/d$ and $e_i \xleftarrow{R} \mathbb{F}_p$ otherwise.
  - Sample $\mathbf{c} \xleftarrow{R} \mathbb{F}_p^k$, $\mathbf{r} \xleftarrow{R} \mathbb{F}_p^\ell$, $s \xleftarrow{R} \mathbb{F}_p$ and compute $\mathbf{z}^\mathsf{T} = \mathbf{r}^\mathsf{T}\mathbf{H} + s\mathbf{c}^\mathsf{T} + \mathbf{e}^\mathsf{T} \in \mathbb{F}_p^k$.

  The algorithm outputs $((\mathbb{G}, p, g), \mathbf{c}, g^{\mathbf{z}})$ as its query and $\mathsf{st} = g^s$ as its state. Here, we write $g^{\mathbf{z}}$ to denote the vector of group elements $(g^{z_1}, \ldots, g^{z_k})$, where $z_1, \ldots, z_k$ are the components of $\mathbf{z}$.
- $\mathcal{P}_{\mathsf{LA}}(q, x, w)$: On input a query $q = ((\mathbb{G}, p, g), \mathbf{c}, g^{\mathbf{z}})$, a GapMDP instance $(\mathbf{A}, d)$ and a witness $\mathbf{v} \in \mathbb{F}_p^k$, the prover algorithm does the following:
  - If $\mathbf{c}^\mathsf{T}\mathbf{v} = 0$, then the prover aborts with output $\perp$. Otherwise, let $t = (\mathbf{c}^\mathsf{T}\mathbf{v})^{-1}$.
  - Output the proof $\pi = g^{t \cdot \mathbf{z}^\mathsf{T}\mathbf{v}}$ (which can be computed from t, $\mathbf{v}$, and $g^{\mathbf{z}}$).
- $\mathcal{V}_{\mathsf{LA}}(\mathsf{st}, \pi)$: On input the verification state $\mathsf{st} \in \mathbb{G}$ and a proof $\pi \in \mathbb{G}$, output 1 if $\mathsf{st} = \pi$, and 0 otherwise.

Completeness and soundness analysis. We now state the completeness and soundness theorems for Construction 5.13 as well as the resulting implication to 1-element laconic arguments and witness encryption for NP in the generic group model. We defer the completeness and soundness proofs to Section 5.3.

Theorem 5.14 (Completeness)**.** *Construction 5.13 has completeness error ε.*

Theorem 5.15 (Soundness)**.** *If $\varepsilon\beta = \omega(\log n)$ and $\mathsf{GroupGen}$ is modeled as a generic group, then Construction 5.13 has soundness error $\mathsf{negl}(\lambda)$.*

Corollary 5.16 (1-Element Laconic Argument for NP)**.** *Let λ be a security parameter. Under Hypothesis 5.12, there exists a predictable laconic argument for NP in the generic group model with completeness error $o(1)$ and soundness error $\mathsf{negl}(\lambda)$ and where the prover's message consists of a single group element.*

Proof. Let $\beta(\lambda) = f(\lambda)\log\lambda$ where $f(\lambda) = \omega(1)$ for which Hypothesis 5.12 holds. Take $\varepsilon = 1/\sqrt{f(\lambda)} = o(1)$. By instantiating Construction 5.13 with this choice of β, ε and appealing to Theorems 5.14 and 5.15, we obtain a laconic argument for GapMDP_β with completeness error $\varepsilon = o(1)$ and soundness error $\mathsf{negl}(\lambda)$. In addition, Construction 5.13 is predictable by construction. Finally, by Hypothesis 5.12, there exists a deterministic polynomial-time Karp-Levin reduction from NP to GapMDP_β, and so we can use our laconic argument for GapMDP_β to obtain a laconic argument for any NP language. □

Corollary 5.17 (Hypothetical Witness Encryption for NP in the Generic Group Model)**.** *Under Hypothesis 5.12, there exists a witness encryption scheme for NP in the generic group model.*

Proof. Follows by instantiating Theorem 5.4 with the predictable laconic argument from Corollary 5.16 (and applying the correctness amplification approach from Remark 5.2). □

Remark 5.18 (Linear PCP for GapMDP)**.** We note that Construction 5.13 implicitly constructs a 2-query (instance-dependent) linear PCP $\Pi_{\mathsf{LPCP}} = (\mathcal{Q}_{\mathsf{LPCP}}, \mathcal{P}_{\mathsf{LPCP}}, \mathcal{D}_{\mathsf{LPCP}})$ for the GapMDP_β problem:

- $\mathcal{Q}_{\mathsf{LPCP}}(\mathbf{A}, d)$: On input a GapMDP_β instance $(\mathbf{A}, d)$ over a finite field $\mathbb{F}$, let $\mathbf{H} \in \mathbb{F}^{\ell \times k}$ be the parity check matrix for the code generated by $\mathbf{A}$. Sample $\mathbf{c} \xleftarrow{R} \mathbb{F}^k$ $\mathbf{r} \xleftarrow{R} \mathbb{F}^\ell$, $\mathbf{e} \in \mathbb{F}^k$ and $s \xleftarrow{R} \mathbb{F}$ as in Construction 5.13. Output the queries $\mathbf{q}_1^\mathsf{T} = \mathbf{r}^\mathsf{T}\mathbf{H} + s\mathbf{c}^\mathsf{T} + \mathbf{e}^\mathsf{T} \in \mathbb{F}^k$ and $\mathbf{q}_2^\mathsf{T} = \mathbf{c}^\mathsf{T} \in \mathbb{F}^k$ and the state $\mathsf{st} = s$.
- $\mathcal{P}_{\mathsf{LPCP}}((\mathbf{A}, d), \mathbf{v})$: On input an instance $(\mathbf{A}, d)$ and a vector $\mathbf{v} \in \mathbb{F}^k$, output the proof $\boldsymbol{\pi} = \mathbf{v} \in \mathbb{F}^k$.

- $\mathcal{V}_{\mathsf{LPCP}}(\mathsf{st}, \mathbf{a})$: On input a state $\mathsf{st} = s$ and responses $\mathbf{a} = (a_1, a_2) \in \mathbb{F}^2$, the verifier accepts if $a_1 \neq 0$ and $a_1 = s \cdot a_2$, and rejects otherwise.

Completeness and soundness of this linear PCP follows via a similar analysis as in the proof of Theorems 5.14 and 5.15.

Remark 5.19 (Witness Encryption via GapMWSP)**.** A natural question is whether we might be able to obtain a predictable argument (and correspondingly, a witness encryption scheme) for NP starting from the 1-query linear PCP for GapMWSP from Construction 4.3. While the 1-query linear PCP in Construction 4.3 is a *predictable* 1-query linear PCP, we do not have a generic compiler that takes a 1-query linear PCP and compiles it into a 1-element argument system (the [BCI+13] compiler with ElGamal encryption yields a 2-element non-predictable argument). Moreover, the general approach from Construction 5.13 of encoding the linear PCP query in the "exponent" does not yield a sound argument system when applied to Construction 4.3. Specifically, the soundness analysis in Construction 5.13 critically relied on the noise vector $\mathbf{e}$ being hidden from the view of the prover. If we directly embed the linear PCP query $\mathbf{r}^\mathsf{T}\mathbf{A} + \mathbf{e}^\mathsf{T}$ from Construction 4.3 in the exponent (i.e., we take the verifier message to be $g^{\mathbf{r}^\mathsf{T}\mathbf{A}+\mathbf{e}^\mathsf{T}}$), then a malicious prover who can find a vector $\mathbf{x} \in \mathbb{F}^n$ with low Hamming weight where $\mathbf{A}\mathbf{x} = \mathbf{0}$ can compute $g^{\mathbf{e}^\mathsf{T}\mathbf{x}}$, which leaks information about the indices of the non-zero components of $\mathbf{e}$. This is not an issue for Construction 5.13 since for false instances, the GapMDP problem stipulates that no such low density vectors $\mathbf{x}$ exist.

Remark 5.20 (Hyperplane Obfuscation and Witness Encryption)**.** An alternative approach to obtaining a witness encryption scheme for the GapMWSP problem is to obfuscate the predictable 1-query linear PCP from Construction 4.3. In more detail, suppose we want to encrypt a message m for the GapMWSP instance $(\mathbf{A}, \mathbf{b}, d)$. To do so, the encrypter first samples a linear PCP query $\mathbf{q} \in \mathbb{F}_p$ for the GapMWSP instance together with the expected response $t \in \mathbb{F}$ by running the query-generation algorithm from Construction 4.3. Then, the encrypter constructs the program $P_{\mathbf{q},t,m}(\boldsymbol{\pi})$ that on input $\boldsymbol{\pi} \in \mathbb{F}^n$ outputs m if $\mathbf{q}^\mathsf{T}\boldsymbol{\pi} = t$ and outputs $\perp$ otherwise. The ciphertext is an obfuscation of the program $P_{\mathbf{q},t,m}$. To decrypt, the decrypter constructs a linear PCP proof $\boldsymbol{\pi} \in \mathbb{F}^n$ for the instance $(\mathbf{A}, \mathbf{b}, d)$ and runs the program on input $\boldsymbol{\pi}$. Correctness of the encryption scheme then follows by completeness of the linear PCP, while semantic security follows from soundness of the linear PCP and assuming that the obfuscated program completely hides $(\mathbf{q}, t, m)$, up to what can be deduced from the input-output behavior of the program $P_{\mathbf{q},t,m}$. For instance if we have an ideal or virtual black-box obfuscation (VBB obfuscation) [BGI+01] of $P_{\mathbf{q},t,m}$, then we can leverage Construction 4.3 to obtain a witness encryption for GapMWSP, and by Lemma 4.2, for all of NP.

While VBB obfuscation for general circuits is impossible in the standard model [BGI+01], we do have positive results for specific families function families such as point functions [Can97, Wee05a, CD08], linear subspaces [CRV10], conjunctions [BVWW16], and more recently, "compute-and-compare" programs [WZ17, GKW17] (also called "lockable obfuscation"). In our setting, the program $P_{\mathbf{q},t,m}$ is simple and corresponds to membership testing in a secret *affine* hyperplane in $\mathbb{F}^n$ (i.e., the set of solutions $\boldsymbol{\pi} \in \mathbb{F}^n$ to the affine relation $\mathbf{q}^\mathsf{T}\boldsymbol{\pi} = t$). More broadly, the program $P_{\mathbf{q},t,m}$ is an example of a compute-and-compare program (i.e., a program that first performs a computation (e.g., $\mathbf{q}^\mathsf{T}\boldsymbol{\pi}$) and outputs a message only if the result of the program matches a target value t). However, existing constructions of lockable obfuscation and compute-and-compare obfuscation require that the target value be computationally unpredictable even given the underlying function. In our case, this means that the target t should be computationally unpredictable even given the query $\mathbf{q}$, which does not hold in our setting. With respect to the obfuscation scheme for hyperplane membership from [CRV10], the construction only support *linear subspaces* (i.e., the case where the target value is 0) and does not extend to affine hyperplane membership testing. Nonetheless, we believe this to be an interesting connection between witness encryption for NP and obfuscation for affine hyperplane membership testing and may provide new avenues for constructing witness encryption for NP (*without* relying on Hypothesis 5.12).

5.3 Analysis of Construction 5.13.

In this section, we give the completeness and soundness analysis of Construction 5.13.

Proof of Theorem 5.14 (Completeness). Let $(\mathbf{A}, d)$ be a GapMDP_β instance with solution $\mathbf{0} \neq \mathbf{v} \in \mathbb{F}_p^k$; namely, $\mathbf{Hv} = \mathbf{0}$ and $\mathsf{wt}(\mathbf{v}) \leq d$. Let $((\mathbb{G}, p, g), \mathbf{c}, g^{\mathbf{z}})$ be the verifier's query. Since $\mathbf{v} \neq \mathbf{0}$ and $\mathbf{c}$ is uniform over $\mathbb{F}_p^k$, we can apply Schwartz-Zippel lemma to conclude that $\mathbf{c}^\mathsf{T}\mathbf{v} \neq 0$ except with probability $1/|\mathbb{F}_p| = \mathsf{negl}(\lambda)$. Thus, the prover can successfully compute $t = (\mathbf{c}^\mathsf{T}\mathbf{v})^{-1}$ and the proof $\pi = g^{t \cdot \mathbf{z}^\mathsf{T}\mathbf{v}}$ from t, $\mathbf{v}$, and $g^{\mathbf{z}}$. By construction,

$$\mathbf{z}^\mathsf{T}\mathbf{v} = \mathbf{r}^\mathsf{T}\mathbf{H}\mathbf{v} + s\mathbf{c}^\mathsf{T}\mathbf{v} + \mathbf{e}^\mathsf{T}\mathbf{v} = s\mathbf{c}^\mathsf{T}\mathbf{v} + \mathbf{e}^\mathsf{T}\mathbf{v},$$

since $\mathbf{Hv} = \mathbf{0}$. Next, we claim that with probability at least $1 - \varepsilon$, $\mathbf{e}^\mathsf{T}\mathbf{v} = 0$. Since $\mathsf{wt}(\mathbf{v}) \leq d$ and each component of $\mathbf{e}$ is zero with probability $1 - \varepsilon/d$, we have that

$$\Pr[\mathbf{e}^\mathsf{T}\mathbf{v} = 0] \geq (1 - \varepsilon/d)^{\mathsf{wt}(\mathbf{v})} \geq (1 - \varepsilon/d)^d \geq 1 - \varepsilon$$

for $d \geq 1$. Thus, with probability at least $1 - \varepsilon$,

$$\pi = g^{t \cdot \mathbf{z}^\mathsf{T}\mathbf{v}} = g^{(\mathbf{c}^\mathsf{T}\mathbf{v})^{-1} s (\mathbf{c}^\mathsf{T}\mathbf{v})} = g^s = \mathsf{st},$$

and the verifier accepts. □

Proof of Theorem 5.15 (Soundness). Take any NO instance $(\mathbf{A}, d)$, and let $\mathbf{H}$ be the parity-check matrix for the code generated by $\mathbf{A}$. We use a hybrid argument:

- Hyb_0: This is the real soundness experiment where we replace the group $(\mathbb{G}, p, g)$ with oracle access to a generic group $\mathcal{G}$. The challenger runs $(\mathsf{pp}, \mathsf{sk}, p) \leftarrow \mathsf{GGM.Setup}(1^\lambda)$. It constructs $\mathbf{z} \in \mathbb{F}_p^k$ as in $\mathcal{Q}_{\mathsf{LA}}$ and computes $h_i \leftarrow \mathsf{GGM.Encode}(\mathsf{sk}, z_i)$, where z_i denotes the i^{th} components of $\mathbf{z}$. It also computes $\mathsf{st} \leftarrow \mathsf{GGM.Encode}(\mathsf{sk}, s)$ and $g \leftarrow \mathsf{GGM.Encode}(\mathsf{sk}, 1)$. The challenger gives $((\mathsf{pp}, p, g), \mathbf{c}, h_1, \ldots, h_k)$ to the prover. The prover is then given access to the generic group oracles $\mathsf{GGM.Setup}$, $\mathsf{GGM.Encode}$, $\mathsf{GGM.Add}$, $\mathsf{GGM.Test}$. At the end of the game, the prover outputs a proof $\pi \in \{0,1\}^\lambda$, and the challenger outputs 1 if π and st encode the same value (by using $\mathsf{GGM.Add}$ and $\mathsf{GGM.Test}$), and 0 otherwise.

- Hyb_1: In this experiment, we change how the generic group oracle queries are implemented. Specifically, the challenger begins by sampling $(\mathsf{pp}, \mathsf{sk}, p) \leftarrow \mathsf{GGM.Setup}(1^\lambda)$ as in Hyb_0. In this experiment, the challenger will explicitly maintain the mapping T of encodings to handles. The challenger samples encodings $g \xleftarrow{\text{R}} \{0,1\}^\lambda$, $h_1, \ldots, h_k \xleftarrow{\text{R}} \{0,1\}^\lambda$ and $\mathsf{st} \xleftarrow{\text{R}} \{0,1\}^\lambda$ and adds mappings $g \mapsto 1$, $h_i \mapsto \hat{z}_i$, $\mathsf{st} \mapsto \hat{s}$, where $\hat{z}_1, \ldots, \hat{z}_k$, and $\hat{s}$ are *formal variables* (see Remark C.3). It also samples $\mathbf{c} \xleftarrow{\text{R}} \mathbb{F}_p^k$ as in Hyb_0. The challenger gives $((\mathsf{pp}, p, g), \mathbf{c}, h_1, \ldots, h_k)$ to the prover. The prover is then given access to the generic group oracles, which are implemented as follows:

  - $\mathsf{GGM.Setup}$: The challenger always replies with $\perp$.
  - $\mathsf{GGM.Encode}$: The challenger always replies with $\perp$.
  - $\mathsf{GGM.Add}$: On input a key k and handles $\xi_1, \xi_2 \in \{0,1\}^\lambda$, the challenger checks that $k = \mathsf{pp}$ and handles ξ_1, ξ_2 are present in T and mapped to formal polynomials f_1, f_2 over $\mathbb{F}_p$ (returning $\perp$ otherwise). If the checks pass, the oracle samples a fresh handle $\xi \xleftarrow{\text{R}} \{0,1\}^\lambda$, adds the entry $\xi \mapsto (f_1 + f_2)$ to T, and replies with ξ.
  - $\mathsf{GGM.Test}$: On input a key k and a handle ξ, the oracle checks that $k = \mathsf{pp}$, that ξ is present in T and mapped to a formal polynomial f over $\mathbb{F}_p$ (returning $\perp$ otherwise). If the checks pass, the oracle outputs "zero" if $f \equiv 0$ is the identically-zero polynomial over $\mathbb{F}_p$ and "nonzero" otherwise.

  At the end of the game, after the prover outputs its proof π, the output of the experiment is 1 if π and st encode identical polynomials over $\mathbb{F}_p$ (i.e., $\pi \mapsto f_1$ and $\mathsf{st} \mapsto f_2$ such that $f_1 - f_2 \equiv 0$ over $\mathbb{F}_p$). Otherwise, the output of the experiment is 0.

- Hyb_2: Same as Hyb_1 except the challenger samples $\mathsf{st} \xleftarrow{\text{R}} \{0,1\}^\lambda$ and adds the mapping $\mathsf{st} \mapsto s$ *after* the adversary has outputted its proof. The output is still computed as in Hyb_1.

For an adversary $\mathcal{A}$, we write $\mathsf{Hyb}_i(\mathcal{A})$ to denote the output of experiment Hyb_i with adversary $\mathcal{A}$.

Lemma 5.21. *For all adversaries $\mathcal{A}$ that make a polynomial number of queries to $\mathcal{G}$, $|\Pr[\mathsf{Hyb}_0(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.*

Proof. First, the components $(\mathsf{pp}, p, g, \mathbf{c}, h_1, \ldots, h_k)$ are identically distributed in Hyb_0 and Hyb_1. Thus, it suffices to show that each of the adversary's queries to the generic group oracle are statistically indistinguishable. We use a hybrid argument over the number of queries the adversary makes, where in the i^{th} hybrid $\mathsf{Hyb}_{0,i}$, the first i queries are answered according to the specification in Hyb_1 while the remaining queries are answered according to the specification in Hyb_0. We show that for all i, the outputs of $\mathsf{Hyb}_{0,i-1}$ and $\mathsf{Hyb}_{0,i}$ are statistically indistinguishable. It suffices then to consider the i^{th} query:

- $\mathsf{GGM.Setup}$: In both $\mathsf{Hyb}_{0,i-1}$ and $\mathsf{Hyb}_{0,i}$, the setup oracle outputs $\perp$.
- $\mathsf{GGM.Encode}$: In Hyb_0, the $\mathsf{GGM.Encode}$ oracle outputs $\perp$ unless the adversary queries the oracle on $k = \mathsf{sk}$. Since the view of $\mathcal{A}$ in the first $i-1$ queries in $\mathsf{Hyb}_{0,i-1}$ is independent of sk and sk is uniform over $\{0,1\}^\lambda$, the output is $\perp$ with probability $1 - 2^{-\lambda}$ (namely, the secret key sk can be sampled *after* the adversary's query). In $\mathsf{Hyb}_{0,i}$, the output of $\mathsf{GGM.Encode}$ on the i^{th} query is always $\perp$, so the output of $\mathsf{GGM.Encode}$ is statistically indistinguishable in the two experiments.
- $\mathsf{GGM.Add}$: The addition oracle has identical behavior in $\mathsf{Hyb}_{0,i-1}$ and $\mathsf{Hyb}_{0,i}$.
- $\mathsf{GGM.Test}$: Suppose an adversary makes a query to the $\mathsf{GGM.Test}$ oracle on $k = \mathsf{pp}$ and a handle $\xi \in \{0,1\}^\lambda$. First, if $k \neq \mathsf{pp}$ or $\xi \notin \mathsf{T}$, then the output in both $\mathsf{Hyb}_{0,i-1}$ and $\mathsf{Hyb}_{0,i}$ is $\perp$, so it suffices to consider the case where $k = \mathsf{pp}$ and $\xi \in \mathsf{T}$. This means that ξ must have been added to T as a result of a $\mathsf{GGM.Encode}$ query or as a result of a $\mathsf{GGM.Add}$ query. By construction, the only values in T from the encoding algorithm correspond to the encodings of 1, $\hat{z}_1, \ldots, \hat{z}_k$, and $\hat{s}$. Any other encodings in T must be the outcome of invoking $\mathsf{GGM.Add}$ on the existing elements. By an inductive argument over the $\mathsf{GGM.Add}$ queries, we can argue that every valid handle ξ in T necessarily corresponds to a linear function in the values 1, $\hat{z}_1, \ldots, \hat{z}_k$, and $\hat{s}$. Thus, any valid zero-test query the adversary makes can be expressed as
$$f(\hat{z}_1, \ldots, \hat{z}_k, \hat{s}) \coloneqq \sum_{i \in [k]} \alpha_i \hat{z}_i + \gamma \hat{s} + \delta,$$
for some choice of scalars $\alpha_1, \ldots, \alpha_k, \gamma, \delta \in \mathbb{F}_p$. If $f \equiv 0$ is the identically-zero polynomial, then the output in both $\mathsf{Hyb}_{0,i-1}$ and $\mathsf{Hyb}_{0,i}$ is "zero." It suffices to consider the case where $f \not\equiv 0$. In this case, the output in $\mathsf{Hyb}_{0,i}$ is always "nonzero," while the output in $\mathsf{Hyb}_{0,i-1}$ depends on the value of f at the concrete values of $z_1, \ldots, z_k, s$ sampled at the beginning of the experiment. Note that by construction, the adversary's view in $\mathsf{Hyb}_{0,i-1}$ in the first $i-1$ queries is independent of the values of $\mathbf{r}$, $\mathbf{e}$, and s. Thus, we can sample the values of $\mathbf{r}, \mathbf{e}, s$ *after* the adversary has chosen its i^{th} query.

  First, let $\boldsymbol{\alpha} \in \mathbb{F}_p^k$ be the vector whose components are $\alpha_1, \ldots, \alpha_k$. Then, we can write
$$f(z_1, \ldots, z_k, s) = \mathbf{z}^\mathsf{T}\boldsymbol{\alpha} + \gamma s + \delta = \mathbf{r}^\mathsf{T}\mathbf{H}\boldsymbol{\alpha} + \mathbf{e}^\mathsf{T}\boldsymbol{\alpha} + (\gamma + \mathbf{c}^\mathsf{T}\boldsymbol{\alpha})s + \delta.$$
  We can re-write $f(z_1, \ldots, z_k, s)$ as a linear polynomial g in the variables $\mathbf{r}$, s, and $\mathbf{e}$:
$$g(\mathbf{r}, s, \mathbf{e}) = \mathbf{r}^\mathsf{T}\mathbf{H}\boldsymbol{\alpha} + \mathbf{e}^\mathsf{T}\boldsymbol{\alpha} + (\gamma + \mathbf{c}^\mathsf{T}\boldsymbol{\alpha})s + \delta = f(z_1, \ldots, z_k, s)$$
  We now consider two cases:

  - Suppose that $\boldsymbol{\alpha} = \mathbf{0}$. In this case, the polynomial $g(\mathbf{r}, s, \mathbf{e}) = \gamma s + \delta$. If g is not identically zero, then at least one of $\gamma, \delta \neq 0$. If $\gamma \neq 0$, this is a nonzero polynomial in s. Since s is uniform over $\mathbb{F}_p$, we apply Schwartz-Zippel to conclude that the probability that $g(\mathbf{r}, s, \mathbf{e}) = 0$ is $1/|\mathbb{F}_p| = \mathsf{negl}(\lambda)$. With overwhelming probability, the zero-test oracle outputs "nonzero." If $\gamma = 0$ and $\delta \neq 0$, then g is a constant nonzero polynomial, and the zero-test oracle always outputs "nonzero."

– Suppose that $\boldsymbol{\alpha} \neq \mathbf{0}$ but $\mathbf{H}\boldsymbol{\alpha} = \mathbf{0}$. Since $(\mathbf{A}, d)$ is a NO instance, it must be the case that $\mathsf{wt}(\boldsymbol{\alpha}) \geq \beta \cdot d$. We argue in this case that $\mathbf{e}^{\mathsf{T}}\boldsymbol{\alpha}$ is not identically-zero with overwhelming probability. By construction, $\Pr[e_i = 0] = 1 - \varepsilon/d$ for each $i \in [k]$, and each component e_i is independent and identically distributed. Let $S_{\boldsymbol{\alpha}} = \{i \in [k] \colon \alpha_i \neq 0\}$ be the set of indices in $\boldsymbol{\alpha}$ that are non-zero. As argued above, $|S_{\boldsymbol{\alpha}}| = \mathsf{wt}(\boldsymbol{\alpha}) \geq \beta \cdot d$. Thus, the probability that $e_i = 0$ for all indices $i \in S_{\boldsymbol{\alpha}}$ is bounded by

$$\Pr[\forall i \in S_{\boldsymbol{\alpha}} : e_i = 0] \leq (1 - \varepsilon/d)^{\mathsf{wt}(\boldsymbol{\alpha})} \leq (1 - \varepsilon/d)^{\beta d} \leq e^{-\beta\varepsilon} = \mathsf{negl}(\lambda),$$

since $\beta\varepsilon = \omega(\log n)$. Thus with overwhelming probability over the choice of $\mathbf{e}$, there exists at least one index i where $e_i \neq 0$ and $\alpha_i \neq 0$. In this case, the polynomial g is a nonzero polynomial in e_i, and since e_i is sampled uniformly over $\mathbb{F}_p$, we apply Schwartz-Zippel to conclude that the probability that $g(\mathbf{r}, s, \mathbf{e}) = 0$ is $1/|\mathbb{F}_p| = \mathsf{negl}(\lambda)$.

– Suppose that $\mathbf{H}\boldsymbol{\alpha} \neq \mathbf{0}$. Then, $g(\mathbf{r}, s, \mathbf{e})$ is a nonzero polynomial in $\mathbf{r}$, and since $\mathbf{r} \xleftarrow{\mathrm{R}} \mathbb{F}_p^{\ell}$, by Schwartz-Zippel, the probability that $g(\mathbf{r}, s, \mathbf{e}) = 0$ is $1/|\mathbb{F}_p| = \mathsf{negl}(\lambda)$.

We conclude that if the adversary queries $\mathsf{GGM.Test}$ on a polynomial that is not identically-zero in the formal variables $\hat{z}_1, \ldots, \hat{z}_k, \hat{s}$, then the output of $\mathsf{GGM.Test}$ in $\mathsf{Hyb}_{0,i-1}$ will be "nonzero" with overwhelming probability, which matches the behavior in $\mathsf{Hyb}_{0,i}$.

The claim now follows by a hybrid argument over the adversary's queries. □

Lemma 5.22. *For all adversaries $\mathcal{A}$ making a polynomial number of queries to $\mathcal{G}$, $|\Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_2(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.*

Proof. The only difference between hybrids Hyb_1 and Hyb_2 is that the challenger defers the sampling of st until the very end. Certainly, the challenge $((\mathsf{pp}, p, g), \mathbf{c}, h_1, \ldots, h_k)$ are identically distributed in the two experiments (they do not depend on st). We take a similar query-by-query approach where st is sampled after the first i queries of the adversary. To show that $\mathsf{Hyb}_{1,i-1}$ and $\mathsf{Hyb}_{1,i}$ are statistically indistinguishable, it suffices to show that the response to i^{th} query is statistically indistinguishable in the two experiments. Since the $\mathsf{GGM.Setup}$ and $\mathsf{GGM.Encode}$ are handled identically in the two experiments, it suffices to consider $\mathsf{GGM.Add}$ and $\mathsf{GGM.Test}$ queries:

- Suppose the i^{th} query is a $\mathsf{GGM.Add}$ query on inputs k, ξ_1, ξ_2. The behavior in the two experiments are identical unless $\xi_1 = \mathsf{st}$ or $\xi_2 = \mathsf{st}$ in $\mathsf{Hyb}_{1,i-1}$. Since st is sampled *after* the first $i - 1$ queries, that means that st is sampled independently of ξ_1 and ξ_2, so over the randomness of st, the probability that $\mathsf{st} = \xi_1$ or $\mathsf{st} = \xi_2$ is $2/2^{\lambda} = \mathsf{negl}(\lambda)$.
- Suppose the i^{th} query is a $\mathsf{GGM.Test}$ query on inputs k, ξ. By the same argument as in the previous case, the behavior is identical unless $\xi = \mathsf{st}$ where st is sampled uniformly and independently of ξ. This happens with probability $1/2^{\lambda} = \mathsf{negl}(\lambda)$ and the claim holds.

The claim now follows by a standard hybrid argument over the adversary's oracle queries. □

Lemma 5.23. *For all adversaries $\mathcal{A}$, $\Pr[\mathsf{Hyb}_2(\mathcal{A}) = 1] = \mathsf{negl}(\lambda)$.*

Proof. Let $\pi \in \{0,1\}^{\lambda}$ be the adversary's output in this experiment. After the adversary outputs π, the challenger samples $\mathsf{st} \xleftarrow{\mathrm{R}} \{0,1\}^{\lambda}$, and with probability $1 - 2^{-\lambda}$, $\pi \neq \mathsf{st}$. Then, there are two possibilities. If $\pi \notin \mathsf{T}$, then the output of the experiment is 0. If $\pi \in \mathsf{T}$, it maps to a polynomial that is independent of $\hat{s}$ (since the only mapping $\mathsf{st} \mapsto \hat{s}$ that involves the formal variable $\hat{s}$ is added *after* the adversary has output π). In this case, π and st cannot correspond to identical polynomials, and the experiment outputs 0. □

The claim now follows by combining Lemmas 5.21 to 5.23. □

Acknowledgments

We thank Nir Bitansky, Benedikt Bünz, Henry Corrigan-Gibbs, Subhash Khot, Sam Kim, and Brent Waters for insightful discussions, comments, and pointers. We thank the anonymous reviewers for helpful feedback on the presentation.

References

[AJL+12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In *EUROCRYPT*, 2012.

[AK14] Per Austrin and Subhash Khot. A simple deterministic reduction for the gap minimum distance of code problem. *IEEE Trans. Inf. Theory*, 60(10), 2014.

[AL07] Yonatan Aumann and Yehuda Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. In *TCC*, 2007.

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3), 1998.

[AP20] Shweta Agrawal and Alice Pellet-Mary. Indistinguishability obfuscation without maps: Attacks and fixes for noisy linear FE. In *EUROCRYPT*, 2020.

[BBB+18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Gregory Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *IEEE SP*, 2018.

[BBC+17] Eli Ben-Sasson, Iddo Bentov, Alessandro Chiesa, Ariel Gabizon, Daniel Genkin, Matan Hamilis, Evgenya Pergament, Michael Riabzev, Mark Silberstein, Eran Tromer, and Madars Virza. Computational integrity with a public random string from quasi-linear PCPs. In *EUROCRYPT*, 2017.

[BBC+19] Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. Zero-knowledge proofs on secret-shared data via fully linear PCPs. In *CRYPTO*, 2019.

[BBFR15] Michael Backes, Manuel Barbosa, Dario Fiore, and Raphael M. Reischuk. ADSNARK: nearly practical and privacy-preserving proofs on authenticated data. In *IEEE SP*, 2015.

[BBHR19] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable zero knowledge with no trusted setup. In *CRYPTO*, 2019.

[BC12] Nir Bitansky and Alessandro Chiesa. Succinct arguments from multi-prover interactive proofs and their efficiency benefits. In *CRYPTO*, 2012.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2), 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 *EUROCRYPT*, 2016.

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

[BCCT12] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *ITCS*, 2012.

[BCG+13] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. SNARKs for C: verifying program executions succinctly and in zero knowledge. In *CRYPTO*, 2013.

[BCG+14] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *IEEE SP*, 2014.

[BCI+13] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In *TCC*, 2013.

[BCPR14] Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. On the existence of extractable one-way functions. In *STOC*, 2014.

[BDGM20a] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Candidate iO from homomorphic encryption schemes. In *EUROCRYPT*, 2020.

[BDGM20b] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Factoring and pairings are not necessary for iO: Circular-secure LWE suffices. *IACR Cryptol. ePrint Arch.*, 2020, 2020.

[BDRV18] Itay Berman, Akshay Degwekar, Ron D. Rothblum, and Prashant Nalini Vasudevan. From laconic zero-knowledge to public-key cryptography - extended abstract. In *CRYPTO*, 2018.

[Ber06] Daniel J. Bernstein. Curve25519: New Diffie-Hellman speed records. In *PKC*, 2006.

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, 2001.

[BHZ87] Ravi B. Boppana, Johan Håstad, and Stathis Zachos. Does co-NP have short interactive proofs? *Inf. Process. Lett.*, 25(2), 1987.

[BIJ+20] James Bartusek, Yuval Ishai, Aayush Jain, Fermi Ma, Amit Sahai, and Mark Zhandry. Affine determinant programs: A framework for obfuscation and witness encryption. In *ITCS*, 2020.

[BISW17] Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu. Lattice-based SNARGs and their application to more efficient obfuscation. In *EUROCRYPT*, 2017.

[BISW18] Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu. Quasi-optimal SNARGs via linear multi-prover interactive proofs. In *EUROCRYPT*, 2018.

[Blo70] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7), 1970.

[BSW12] Dan Boneh, Gil Segev, and Brent Waters. Targeted malleability: homomorphic encryption for restricted computations. In *ITCS*, 2012.

[BVWW16] Zvika Brakerski, Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Obfuscating conjunctions under entropic ring LWE. In *ITCS*, 2016.

[Can97] Ran Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In *CRYPTO*, 1997.

[CD08] Ran Canetti and Ronny Ramzi Dakdouk. Obfuscating point functions with multibit output. In *EUROCRYPT*, 2008.

[CDM+18] Geoffroy Couteau, Aurélien Dupin, Pierrick Méaux, Mélissa Rossi, and Yann Rotella. On the concrete security of goldreich's pseudorandom generator. In *ASIACRYPT*, 2018.

[CL08] Giovanni Di Crescenzo and Helger Lipmaa. Succinct NP proofs from an extractability assumption. In *Conference on Computability in Europe*, 2008.

[CO99] Ran Canetti and Rafail Ostrovsky. Secure computation with honest-looking parties: What if nobody is truly honest? In *STOC*, 1999.

[CRV10] Ran Canetti, Guy N. Rothblum, and Mayank Varia. Obfuscation of hyperplane membership. In *TCC*, 2010.

[CVW18] Yilei Chen, Vinod Vaikuntanathan, and Hoeteck Wee. GGH15 beyond permutation branching programs: Proofs, attacks, and candidates. In *CRYPTO*, 2018.

[CW09] Qi Cheng and Daqing Wan. A deterministic reduction for the gap minimum distance problem: [extended abstract]. In *STOC*, 2009.

[DFGK14] George Danezis, Cédric Fournet, Jens Groth, and Markulf Kohlweiss. Square span programs with applications to succinct NIZK arguments. In *ASIACRYPT*, 2014.

[DMS99] Ilya Dumer, Daniele Micciancio, and Madhu Sudan. Hardness of approximating the minimum distance of a linear code. In *FOCS*, 1999.

[DS14] Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *STOC*, 2014.

[ElG84] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO*, 1984.

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In *CRYPTO*, 2018.

[FNV17] Antonio Faonio, Jesper Buus Nielsen, and Daniele Venturi. Predictable arguments of knowledge. In *PKC*, 2017.

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, 2013.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct nizks without pcps. In *EUROCRYPT*, 2013.

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In *STOC*, 2013.

[GH98] Oded Goldreich and Johan Håstad. On the complexity of interactive proofs with bounded communication. *Inf. Process. Lett.*, 67(4), 1998.

[GJLS20] Romain Gay, Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from simple-to-state hard problems: New assumptions, new techniques, and simplification. *IACR Cryptol. ePrint Arch.*, 2020, 2020.

[GKW17] Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In *FOCS*, 2017.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *STOC*, 1989.

[GLW14] Craig Gentry, Allison B. Lewko, and Brent Waters. Witness encryption from instance independent assumptions. In *CRYPTO*, 2014.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *STOC*, 1985.

[Gol00] Oded Goldreich. Candidate one-way functions based on expander graphs. *Electronic Colloquium on Computational Complexity (ECCC)*, 7(90), 2000.

[GP20] Romain Gay and Rafael Pass. Indistinguishability obfuscation from circular security. *IACR Cryptol. ePrint Arch.*, 2020, 2020.

[Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In *ASIACRYPT*, 2010.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In *EUROCRYPT*, 2016.

[GVW01] Oded Goldreich, Salil P. Vadhan, and Avi Wigderson. On interactive proofs with a laconic prover. In *ICALP*, 2001.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *STOC*, 2011.

[Hås01] Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4), 2001.

[Hel80] Martin E. Hellman. A cryptanalytic time-memory trade-off. *IEEE Trans. Information Theory*, 26(4), 1980.

[HKLT19] Prahladh Harsha, Subhash Khot, Euiwoong Lee, and Devanathan Thiruvenkatachari. Improved 3lin hardness via linear label cover. In *APPROX/RANDOM*, 2019.

[Hoe63] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301), 1963.

[IKO07] Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. Efficient arguments without short PCPs. In *CCC*, 2007.

[IR89] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *STOC*, 1989.

[JLS20] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. *IACR Cryptol. ePrint Arch.*, 2020, 2020.

[Kho20] Subhash Khot. Personal communication, 2020.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments. In *STOC*, 1992.

[KLM+18] Sam Kim, Kevin Lewi, Avradip Mandal, Hart Montgomery, Arnab Roy, and David J. Wu. Function-hiding inner product encryption is practical. In *SCN*, 2018.

[KMO89] Joe Kilian, Silvio Micali, and Rafail Ostrovsky. Minimum resource zero-knowledge proofs. In *FOCS*, 1989.

[KPV12] Subhash Khot, Preyas Popat, and Nisheeth K. Vishnoi. $2^{\log 1\text{-}\varepsilon\ n}$ hardness for the closest vector problem with preprocessing. In *STOC*, 2012.

[Lip12] Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In *TCC*, 2012.

[Mic00] Silvio Micali. Computationally sound proofs. *SIAM J. Comput.*, 30(4), 2000.

[Mie08] Thilo Mie. Polylogarithmic two-round argument systems. *J. Mathematical Cryptology*, 2(4), 2008.

[MMN+16a] Mohammad Mahmoody, Ameer Mohammed, Soheil Nematihaji, Rafael Pass, and abhi shelat. Lower bounds on assumptions behind indistinguishability obfuscation. In *TCC*, 2016.

[MMN+16b] Mohammad Mahmoody, Ameer Mohammed, Soheil Nematihaji, Rafael Pass, and abhi shelat. A note on black-box complexity of indistinguishability obfuscation. *IACR Cryptol. ePrint Arch.*, 2016, 2016.

[MR10] Dana Moshkovitz and Ran Raz. Two-query PCP with subconstant error. *J. ACM*, 57(5), 2010.

[Nec94] V.I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *MATHEMATICAL NOTES*, 55, 1994.

[PHGR13] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *IEEE SP*, 2013.

[Pol78] John M Pollard. Monte carlo methods for index computation (mod p). *Mathematics of computation*, 32(143), 1978.

[PRV12] Periklis A. Papakonstantinou, Charles Rackoff, and Yevgeniy Vahlis. How powerful are the DDH hard groups? *IACR Cryptol. ePrint Arch.*, 2012, 2012.

[Raz95] Ran Raz. A parallel repetition theorem. In *STOC*, 1995.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, 2005.

[Sch80] Jacob T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4), 1980.

[SCI20] SCIPR Lab. libsnark: a C++ library for zkSNARK proofs. https://github.com/scipr-lab/libsnark, 2020.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT*, 1997.

[Wee05a] Hoeteck Wee. On obfuscating point functions. In *STOC*, 2005.

[Wee05b] Hoeteck Wee. On round-efficient argument systems. In *ICALP*, 2005.

[WW20] Hoeteck Wee and Daniel Wichs. Candidate obfuscation via oblivious LWE sampling. *IACR Cryptol. ePrint Arch.*, 2020, 2020.

[WZ17] Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under LWE. In *FOCS*, 2017.

[Zim15] Joe Zimmerman. How to obfuscate programs directly. In *EUROCRYPT*, 2015.

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *EUROSAM*, 1979.

A Succinct Non-Interactive Arguments and Laconic Arguments

We recall the definitions of a succinct non-interactive argument (SNARG) and a two-message laconic argument for arithmetic circuit satisfiability (which includes Boolean circuit satisfiability as a special case).

Definition A.1 (Succinct Non-Interactive Argument)**.** Let $\mathcal{C} = \{C_\ell\}_{\ell \in \mathbb{N}}$ be a family of arithmetic circuits and let $\mathcal{R}_\mathcal{C}$ be the corresponding circuit satisfiability relation (and let $\mathcal{L}_\mathcal{C}$ be the associated language). A succinct non-interactive argument (SNARG) for $\mathcal{R}_\mathcal{C}$ with completeness error $c = c(\lambda)$ and soundness error $\varepsilon = \varepsilon(\lambda)$ is a tuple $\Pi_\mathsf{SNARG} = (\mathcal{S}_\mathsf{SNARG}, \mathcal{P}_\mathsf{SNARG}, \mathcal{V}_\mathsf{SNARG})$ with the following syntax:

- $\mathcal{S}_\mathsf{SNARG}(1^\lambda) \to (\mathsf{crs}, \mathsf{st})$: On input the security parameter λ, the setup algorithm outputs a common reference string crs and verification state st.

- $\mathcal{P}_{\mathsf{SNARG}}(\mathsf{crs}, x, w) \to \pi$: On input a common reference string crs, a statement x, and a witness w, the prove algorithm outputs a proof π.
- $\mathcal{V}_{\mathsf{SNARG}}(\mathsf{st}, x, \pi) \to \{0,1\}$: On input the verification state st, a statement x and a proof π, the verification algorithm outputs a bit $b \in \{0,1\}$.

Moreover, Π_{SNARG} should satisfy the following properties:

- **Completeness:** For all $\lambda \in \mathbb{N}$ and all $(x, w) \in \mathcal{R}_{\mathcal{C}}$,

$$\Pr[\mathcal{V}_{\mathsf{SNARG}}(\mathsf{st}, x, \pi) = 1 \mid (\mathsf{crs}, \mathsf{st}) \leftarrow \mathcal{S}_{\mathsf{SNARG}}(1^\lambda), \pi \leftarrow \mathcal{P}_{\mathsf{SNARG}}(\mathsf{crs}, x, w)] \geq 1 - c(\lambda).$$

  When $c(\lambda) = \mathsf{negl}(\lambda)$, we say that Π_{SNARG} provides statistical completeness and when $c(\lambda) = 0$, we say that Π_{SNARG} provides perfect completeness.

- **Soundness:** We say that Π_{SNARG} is *non-adaptively* sound if for all $\lambda \in \mathbb{N}$, and all efficient provers $\mathcal{P}^*$, and all instances $x \notin \mathcal{L}_{\mathcal{C}}$,

$$\Pr[\mathcal{V}_{\mathsf{SNARG}}(\mathsf{st}, x, \pi) = 1 \mid (\mathsf{crs}, \mathsf{st}) \leftarrow \mathcal{S}_{\mathsf{SNARG}}(1^\lambda), \pi \leftarrow \mathcal{P}^*(1^\lambda, \mathsf{crs}, x)] \leq \varepsilon(\lambda).$$

  We say that Π_{SNARG} is *adaptively* sound if for all $\lambda \in \mathbb{N}$, and all efficient provers $\mathcal{P}^*$, we have that

$$\Pr[\mathcal{V}_{\mathsf{SNARG}}(\mathsf{st}, x, \pi) = 1 \wedge x \notin \mathcal{L}_{\mathcal{C}} \mid (\mathsf{crs}, \mathsf{st}) \leftarrow \mathcal{S}_{\mathsf{SNARG}}(1^\lambda), (x, \pi) \leftarrow \mathcal{P}^*(1^\lambda, \mathsf{crs})] \leq \varepsilon(\lambda).$$

- **Optimal preprocessing succinctness:** There exists universal polynomials p_1, p_2 (independent of $\mathcal{C}$) such that $\mathcal{S}_{\mathsf{SNARG}}$ and $\mathcal{P}_{\mathsf{SNARG}}$ run in time $p_1(\lambda, |C_\ell|)$, $\mathcal{V}_{\mathsf{SNARG}}$ runs in time $o(|C_\ell|) \cdot p_2(\lambda, |x|)$, and the proof size is $O(\lambda)$.

Remark A.2 (Polylogarithmic Verification Complexity)**.** Some definitions of succinct arguments impose a more stringent efficiency requirement that the verification complexity is *polylogarithmic* in the size $|C_\ell|$ of the classic NP verifier. Our focus in this work is minimizing the proof size, so our only requirement on the verification complexity is that it should be sublinear in the size of the classic NP verifier (similar to the original definition from [GW11]).

Remark A.3 (Public vs. Designated Verifier)**.** We say a SNARG is *publicly-verifiable* if the verification state $\mathsf{st} = \perp$ (namely, anyone can verify in this case). We say a SNARG is in the designated-verifier model if only the holder of st is able to verify proofs (i.e., $\mathcal{V}_{\mathsf{SNARG}}$ takes st as input).

2-message laconic arguments. A 2-message laconic argument can be viewed as a SNARG with a "statement-dependent" common reference string which the verifier sends to the prover in the first round. The remaining properties are defined analogously to Definition A.1. We give the full definition below:

Definition A.4 (2-Message Laconic Argument)**.** Let $\mathcal{C} = \{C_\ell\}_{\ell \in \mathbb{N}}$ be a family of arithmetic circuits, and let $\mathcal{R}_{\mathcal{C}}$ be the corresponding circuit satisfiability relation (and let $\mathcal{L}_{\mathcal{C}}$ be the associated language). A 2-message laconic argument for $\mathcal{R}_{\mathcal{C}}$ with completeness error $c = c(\lambda)$ and soundness error $\varepsilon = \varepsilon(\lambda)$ is a tuple $\Pi_{\mathsf{LA}} = (\mathcal{Q}_{\mathsf{LA}}, \mathcal{P}_{\mathsf{LA}}, \mathcal{V}_{\mathsf{LA}})$ with the following syntax:

- $\mathcal{Q}_{\mathsf{LA}}(1^\lambda, x) \to (q, \mathsf{st})$: On input the security parameter λ and a statement x, the query algorithm outputs a query q and a verification state st.
- $\mathcal{P}_{\mathsf{LA}}(q, x, w) \to \pi$: On input a query q, a statement x, and a witness w, the prove algorithm outputs a proof π.
- $\mathcal{V}_{\mathsf{LA}}(\mathsf{st}, \pi) \to \{0,1\}$: On input the verification state st and a proof π, the verification algorithm outputs a bit $b \in \{0,1\}$.

Moreover, $\Pi_{\mathsf{LA}} = (\mathcal{Q}_{\mathsf{LA}}, \mathcal{P}_{\mathsf{LA}}, \mathcal{V}_{\mathsf{LA}})$ should satisfy the following properties:

- **Completeness:** For all $\lambda \in \mathbb{N}$ and all $(x, w) \in \mathcal{R}_{\mathcal{C}}$,

$$\Pr[\mathcal{V}_{\mathsf{LA}}(\mathsf{st}, \pi) = 1 \mid (q, \mathsf{st}) \leftarrow \mathcal{Q}_{\mathsf{LA}}(1^\lambda, x), \pi \leftarrow \mathcal{P}_{\mathsf{LA}}(q, x, w)] \geq 1 - c(\lambda).$$

  If $c(\lambda) = \mathsf{negl}(\lambda)$, then we say Π_{LA} is statistically correct and if $c(\lambda) = 0$, then we say that Π_{LA} is perfectly correct.

- **Soundness:** For all $\lambda \in \mathbb{N}$ and all efficient provers $\mathcal{P}^*$, and all instances $x \notin \mathcal{L}_{\mathcal{C}}$,

$$\Pr[\mathcal{V}_{\mathsf{LA}}(\mathsf{st}, \pi) = 1 \mid (q, \mathsf{st}) \leftarrow \mathcal{Q}_{\mathsf{LA}}(1^\lambda, x), \pi \leftarrow \mathcal{P}^*(1^\lambda, q, x)] \leq \varepsilon(\lambda).$$

- **Optimal succinctness:** There exist universal polynomials p_1, p_2 (independent of $\mathcal{C}$) such that $\mathcal{Q}_{\mathsf{LA}}$ and $\mathcal{P}_{\mathsf{LA}}$ run in time $p_1(\lambda, |C_\ell|)$, $\mathcal{V}_{\mathsf{LA}}$ runs in time $o(|C_\ell|) \cdot p_2(\lambda, |x|)$ and the length of the proof output by $\mathcal{P}_{\mathsf{LA}}$ is $O(\lambda)$.

A.1 The Bitansky et al. Compiler

In this section, we recall the compiler of Bitansky et al. [BCI+13] from linear PCPs to preprocessing SNARGs. The general compiler proceeds in two main steps:

- The first step of the compiler constructs a 2-message linear interactive proof (LIP) from a linear PCP. At a high level, a linear interactive proof is a standard interactive proof [GMR85], except that each message from the prover to the verifier is a linear (or more generally, affine) function of the verifier's messages. We refer to [BCI+13, §2.3] for the formal definition. Bitansky et al. showed how to compile any k-query linear PCP into a 2-message LIP by introducing a linear consistency check.

- The second step of the compiler takes a 2-message LIP and constructs from it a preprocessing SNARG using a notion called linear-only encryption (i.e., an encryption scheme that only supports linear homomorphism). The approach here is to encrypt the verifier's query in the LIP using the linear-only encryption scheme, and have the prover homomorphically compute its response to the encrypted queries. Here, the linear-only property constrains the prover to only implement linear (or more generally, affine) strategies, thus cryptographically enforcing the constraints of the LIP model. This step can also be instantiated from weaker notions of linear-only such as linear targeted malleability [BSW12]. In the case where the verifier's message in the LIP is statement-independent, the verifier's encrypted queries can be generated in advance (as part of the common reference string); this yields a preprocessing SNARG. Conversely, if the verifier's message in the LIP is statement-dependent, then the resulting construction yields a 2-message laconic argument. In Appendix C.1, we show that the ElGamal encryption scheme over a group $\mathbb{G}$ (see Construction C.4) satisfies the necessary notion of linear targeted malleability to instantiate the [BCI+13] compiler when the group $\mathbb{G}$ is modeled as a generic group.

As noted in [BCI+13, Remark 2.8], a linear PCP with one query is already a 2-message LIP, where the prover's message consists of a single field element. Thus, we can directly apply the second step of the [BCI+13] compiler to a 1-query linear PCP without needing to first construct a linear interactive proof. Below, we give the formal statements of the [BCI+13] compiler starting from 1-query linear PCPs (as this is the only setting we require in this work). We start by recalling the definition of linear targeted malleability.

Definition A.5 (Additively Homomorphic Encryption)**.** An additively homomorphic encryption scheme over a ring $\mathbb{Z}_q$ is a tuple of algorithms $\Pi_{\mathsf{Enc}} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Add}, \mathsf{ImVer})$ with the following syntax:

- $\mathsf{Setup}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$: On input the security parameter λ, the setup algorithm outputs a public key pk and a secret key sk.

- $\mathsf{Encrypt}(\mathsf{pk}, m) \to \mathsf{ct}$: On input the public key pk and a message $m \in \mathbb{Z}_q$, the encryption algorithm outputs a ciphertext ct.

- $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}) \to m$: On input the secret key sk and a ciphertext ct, the decryption algorithm outputs a message $m \in \mathbb{Z}_q \cup \{\perp\}$.
- $\mathsf{Add}(\mathsf{pk}, \mathsf{ct}_1, \mathsf{ct}_2) \to \mathsf{ct}'$: On input the public key pk and two ciphertexts $\mathsf{ct}_1, \mathsf{ct}_2$, the addition algorithm outputs a new ciphertext ct'.
- $\mathsf{ImVer}(\mathsf{sk}, \mathsf{ct}) \to \{0,1\}$: On input the secret key sk and a ciphertext ct, the image-verification algorithm outputs a bit $b \in \{0,1\}$.

Moreover, an additively homomorphic encryption scheme should satisfy the following properties:

- **Correctness:** For every $\lambda \in \mathbb{N}$ and every message $m \in \mathbb{Z}_q$,
$$\Pr\left[\mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}) = m \wedge \mathsf{ImVer}(\mathsf{sk}, \mathsf{ct}) = 1 \;\middle|\; \begin{array}{c}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda);\\ \mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{pk}, m)\end{array}\right] = 1.$$
- **Additive homomorphism:** For every $\lambda \in \mathbb{N}$, every $(\mathsf{pk}, \mathsf{sk})$ in the support of $\mathsf{Setup}(1^\lambda)$, and all ciphertexts $\mathsf{ct}_1, \mathsf{ct}_2$ in the support of $\mathsf{Encrypt}(\mathsf{pk}, m_1)$ and $\mathsf{Encrypt}(\mathsf{pk}, m_2)$, respectively, where $m_1, m_2 \in \mathbb{Z}_q$,
$$\Pr[\mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}') = m_1 + m_2 \wedge \mathsf{ImVer}(\mathsf{sk}, \mathsf{ct}') = 1 \mid \mathsf{ct}' \leftarrow \mathsf{Add}(\mathsf{pk}, \mathsf{ct}_1, \mathsf{ct}_2)] = 1.$$
- **Semantic security:** For any (sufficiently large) $\lambda \in \mathbb{N}$, and every efficient and stateful adversary $\mathcal{A}$,
$$\Pr\left[b = b' \;\middle|\; \begin{array}{c}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda), (m_0, m_1) \leftarrow \mathcal{A}(1^\lambda, \mathsf{pk});\\ b \xleftarrow{R} \{0,1\}, \mathsf{ct}_b \leftarrow \mathsf{Encrypt}(\mathsf{pk}, m_b);\\ b' \leftarrow \mathcal{A}(\mathsf{ct}_b)\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

In our case, we allow $\mathsf{Decrypt}$ and ImVer to additionally take as input a set $\mathcal{M} \subseteq \mathbb{Z}_q$ of potential messages, and we only require the first two properties to hold for ciphertexts encrypting messages in $\mathcal{M}$:

- **Correctness:** For every $\lambda \in \mathbb{N}$, every set $\mathcal{M} \subseteq \mathbb{Z}_q$, and every message $m \in \mathcal{M}$,
$$\Pr\left[\mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}, \mathcal{M}) = m \wedge \mathsf{ImVer}(\mathsf{sk}, \mathsf{ct}, \mathcal{M}) = 1 \;\middle|\; \begin{array}{c}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda);\\ \mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{pk}, m)\end{array}\right] = 1.$$
- **Additive homomorphism:** For every $\lambda \in \mathbb{N}$, every set $\mathcal{M} \subseteq \mathbb{Z}_q$, every $(\mathsf{pk}, \mathsf{sk})$ in the support of $\mathsf{Setup}(1^\lambda)$, and every ciphertexts $\mathsf{ct}_1, \mathsf{ct}_2$ in the support of $\mathsf{Encrypt}(\mathsf{pk}, m_1)$ and $\mathsf{Encrypt}(\mathsf{pk}, m_2)$, respectively, where $m_1, m_2 \in \mathbb{Z}_q$, if $m_1 + m_2 \in \mathcal{M}$, then
$$\Pr[\mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}', \mathcal{M}) = m_1 + m_2 \wedge \mathsf{ImVer}(\mathsf{sk}, \mathsf{ct}', \mathcal{M}) = 1 \mid \mathsf{ct}' \leftarrow \mathsf{Add}(\mathsf{pk}, \mathsf{ct}_1, \mathsf{ct}_2)] = 1.$$

Definition A.6 (Linear Targeted Malleability [BSW12, adapted])**.** An additively homomorphic encryption scheme $\Pi_{\mathsf{Enc}} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Add}, \mathsf{ImVer})$ over $\mathbb{Z}_q$ satisfies *linear targeted malleability* if for every efficient adversary $\mathcal{A}$ and plaintext generator $\mathcal{T}$, there is an efficient simulator $\mathcal{S}$ such that for any sufficiently large $\lambda \in \mathbb{N}$ and auxiliary input $z \in \{0,1\}^{\mathsf{poly}(\lambda)}$, the outputs of the following two distributions are computationally indistinguishable:

Real Distribution:
1. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$
2. $(s, a_1, \ldots, a_m) \leftarrow \mathcal{T}(\mathsf{pk})$
3. $\mathsf{ct}_i \leftarrow \mathsf{Encrypt}(\mathsf{pk}, a_i)$ for all $i \in [m]$
4. $\mathsf{ct}'_1, \ldots, \mathsf{ct}'_k \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_m; z)$
5. $a'_i \leftarrow \mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}')$ for all $i \in [k]$
6. If $\mathsf{ImVer}(\mathsf{ct}'_i) = 1$ for all $i \in [k]$, output $\big(\mathsf{pk}, \{a_i\}_{i\in[m]}, s, \{a'_i\}_{i\in[k]}\big)$.
7. Otherwise, output $\perp$.

Ideal Distribution:
1. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$
2. $(s, a_1, \ldots, a_m) \leftarrow \mathcal{T}(\mathsf{pk})$
3. $\mathsf{ct}_i \leftarrow \mathsf{Encrypt}(\mathsf{pk}, a_i)$ for all $i \in [m]$
4. $\mathsf{ct}'_1, \ldots, \mathsf{ct}'_k \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{ct}_1, \ldots, \mathsf{ct}_m; z)$
5. $(\mathbf{\Pi}, \mathbf{b}) \leftarrow \mathcal{S}(\mathsf{pk}; z)$
6. $(a'_1, \ldots, a'_k)^\mathsf{T} \leftarrow \mathbf{\Pi} \cdot (a_1, \ldots, a_m)^\mathsf{T} + \mathbf{b}$
7. If $\mathsf{ImVer}(\mathsf{ct}'_i) = 1$ for all $i \in [k]$, output $\big(\mathsf{pk}, \{a_i\}_{i\in[m]}, s, \{a'_i\}_{i\in[k]}\big)$.
8. Otherwise, output $\perp$.

In our setting, we will also consider a restricted version of linear targeted malleability where we require the above distributions to be indistinguishable only when the ciphertexts produced by the adversary correspond to messages drawn from a restricted subset $\mathcal{M} \subseteq \mathbb{Z}_q$. In this case, we augment the above definition by also including the set $\mathcal{M}$ as an input to $\mathsf{Decrypt}$ and ImVer (Definition A.5). In this case, we say that Π_{Enc} satisfies *linear targeted malleability with respect to the target message space* $\mathcal{M}$.

Remark A.7 (Auxiliary Input Distributions)**.** Definition A.6 requires the simulator to succeed for arbitrary auxiliary inputs $z \in \{0,1\}^{\mathsf{poly}(\lambda)}$. This requirement is quite strong since z can be used to encode difficult cryptographic problems that the simulator needs to solve in order to correctly simulate the output distribution [BCPR14]. However, in many scenarios, it suffices to just consider "benign" distributions for which the definition plausibly holds. For instance, in our application to preprocessing SNARGs and laconic arguments, it suffices to consider the setting where the auxiliary input z is a uniform string.

Preprocessing SNARGs from linear PCPs. We now recall the construction of preprocessing SNARGs from any 1-query linear PCP in conjunction with an additively homomorphic encryption scheme satisfying linear targeted malleability.

Construction A.8 (SNARG from 1-Query Linear PCP [BCI+13])**.** Let $\mathcal{C} = \{C_\ell\}_{\ell \in \mathbb{N}}$ be a family of arithmetic circuits over a finite field $\mathbb{F}$ and let $\mathcal{R}_{\mathcal{C}}$ be the associated circuit satisfiability relation. Let $\Pi_{\mathsf{LPCP}} = (\mathcal{Q}_{\mathsf{LPCP}}, \mathcal{P}_{\mathsf{LPCP}}, \mathcal{D}_{\mathsf{LPCP}})$ be a 1-query input-oblivious linear PCP for $\mathcal{R}_{\mathcal{C}}$ over $\mathbb{F}$, and let $\Pi_{\mathsf{Enc}} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Add}, \mathsf{ImVer})$ be an additively homomorphic encryption scheme over $\mathbb{F}$. We construct a SNARG $\Pi_{\mathsf{SNARG}} = (\mathcal{S}_{\mathsf{SNARG}}, \mathcal{P}_{\mathsf{SNARG}}, \mathcal{V}_{\mathsf{SNARG}})$ for $\mathcal{R}_{\mathcal{C}}$ as follows:

- $\mathcal{S}_{\mathsf{SNARG}}(1^\lambda)$: On input the security parameter λ, the setup algorithm runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and $(\mathsf{st}', \mathbf{q}_{\mathsf{inp}}, \mathbf{q}) \leftarrow \mathcal{Q}_{\mathsf{LPCP}}$ where $\mathbf{q}_{\mathsf{inp}} \in \mathbb{F}^n$ and $\mathbf{q} = (q_1, \ldots, q_\ell) \in \mathbb{F}^\ell$. Then, the setup algorithm computes the ciphertexts $\mathsf{ct}_i \leftarrow \mathsf{Encrypt}(\mathsf{pk}, q_i)$ for all $i \in [\ell]$. The setup algorithm outputs the CRS $\mathsf{crs} = (\mathsf{pk}, \{\mathsf{ct}_i\}_{i \in [\ell]})$ and the secret verification state $\mathsf{st} = (\mathsf{sk}, \mathsf{st}', \mathbf{q}_{\mathsf{inp}})$.
- $\mathcal{P}_{\mathsf{SNARG}}(\mathsf{crs}, \mathbf{x}, \mathbf{w})$: On input a common reference string $\mathsf{crs} = \big(\mathsf{pk}, \{\mathsf{ct}_i\}_{i \in [\ell]}\big)$, a statement $\mathbf{x} \in \mathbb{F}^n$, and a witness $\mathbf{w}$, the prover computes a proof $\boldsymbol{\pi} \leftarrow \mathcal{P}_{\mathsf{LPCP}}(\mathbf{x}, \mathbf{w}) \in \mathbb{F}^\ell$. It interprets $\{\mathsf{ct}_i\}_{i \in [\ell]}$ as the encryption of a vector $\mathbf{q} \in \mathbb{F}^\ell$ and homomorphically computes an encryption ct' of the value $\mathbf{q}^\mathsf{T}\boldsymbol{\pi}$. It outputs the proof $\pi = \mathsf{ct}'$.
- $\mathcal{V}_{\mathsf{SNARG}}(\mathsf{st}, \mathbf{x}, \pi)$: On input the verification state $\mathsf{st} = (\mathsf{sk}, \mathsf{st}', \mathbf{q}_{\mathsf{inp}})$, the statement $\mathbf{x} \in \mathbb{F}^n$, and a proof $\pi = \mathsf{ct}'$, the verifier first checks that $\mathsf{ImVer}(\mathsf{sk}, \mathsf{ct}') = 1$ and outputs 0 if not. Otherwise, it computes $a \leftarrow \mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}')$ and outputs $\mathcal{D}_{\mathsf{LPCP}}(\mathsf{st}', \mathbf{q}_{\mathsf{inp}}^\mathsf{T}\mathbf{x}, a)$.

Theorem A.9 (Preprocessing SNARG from 1-Query Linear PCP [BCI+13], adapted)**.** *If* Π_{LPCP} *has soundness error* ε *and* Π_{Enc} *satisfies linear targeted malleability, then Construction A.8 gives a preprocessing SNARG for* $\mathcal{R}_{\mathcal{C}}$ *with perfect completeness and (non-adaptive) soundness error* $\varepsilon + \mathsf{negl}(\lambda)$.

Remark A.10 (Adaptive Soundness)**.** Previously, Bitansky et al. [BCI+13] showed that if the underlying encryption scheme Π_{Enc} in Construction A.8 satisfies the stronger extractability notion of linear-only security, then the resulting SNARG satisfies *adaptive* soundness (where the malicious prover can choose the statement to prove *after* seeing the CRS). In addition, we note that some *specific* instantiations of Construction A.8 can also be shown to satisfy adaptive soundness without necessarily requiring Π_{Enc} satisfy the linear-only property. For example, we show in Appendix C.2 that Construction A.8 provides adaptive soundness when Π_{Enc} is instantiated with the ElGamal encryption scheme over a group $\mathbb{G}$ that is modeled as a *generic group*. In this setting of adaptively-sound SNARGs, the Gentry-Wichs lower bound [GW11] rules out the possibility of basing security on a non-falsifiable assumption. As such, some kind of non-falsifiable assumption (e.g., the linear-only assumption) or working in an idealized model like the generic group model seems necessary for the security analysis.

Laconic arguments from linear PCPs. If the underlying linear PCP in Construction A.8 is *input-dependent*, then we obtain a SNARG where the CRS is statement-dependent—that is, a 2-message laconic argument. Thus, we can state the corresponding analog of Theorem A.9 for laconic arguments:

Theorem A.11 (Laconic Arguments from 1-Query Linear PCPs [BCI+13, adapted])**.** *If* Π_{LPCP} *is an* input-dependent *linear PCP with soundness error* ε *and* Π_{Enc} *satisfies linear targeted malleability, then Construction A.8 yields a 2-message laconic argument for* $\mathcal{R}_{\mathcal{C}}$ *with perfect completeness and soundness error* $\varepsilon + \mathsf{negl}(\lambda)$.

Remark A.12 (Bounded Linear PCPs)**.** When a 1-query linear PCP has the property that all *honestly-generated* proofs lie in a set $\mathcal{M} \subseteq \mathbb{F}$ known to the verifier (e.g., $\mathcal{M}$ may be determined by the query-generation algorithm), we can tweak Construction A.8 so that Theorems A.9 and A.11 hold as long the encryption scheme Π_{Enc} in Construction A.8 satisfy linear targeted malleability with respect to $\mathcal{M}$ (Definition A.6). Namely, we modify the verification algorithm to additionally include the set $\mathcal{M}$ as input to ImVer and $\mathsf{Decrypt}$. This means that the SNARG verifier only accepts if the prover's response is an encryption of a message in the set $\mathcal{M}$, and moreover, every prover strategy that convinces the verifier to accept can be explained by an affine function of the verifier's encrypted queries. Thus, completeness and soundness follow by the same argument as in the proofs of Theorems A.9 and A.11 from [BCI+13]. An immediate consequence is that if we have a bounded linear PCP where honestly-generated responses lie in a polynomial-size subset $\mathcal{M} \subseteq |\mathbb{F}|$, we can obtain a SNARG (or laconic argument) from an encryption scheme that satisfies linearly targeted malleability with respect to polynomial-size subsets (e.g., the ElGamal encryption scheme (Construction C.4 and Theorem C.6)).

B Linear PCPs for Boolean Circuit Satisfiability

In this section, we review the classic construction of a linear PCP based on the Walsh-Hadamard code (following the presentation from [BCI+13, BISW17]). We then show how to modify it to obtain a bounded 2-query linear PCP with either strong soundness or δ-honest verifier zero knowledge for the language of Boolean circuit satisfiability.

B.1 A Linear PCP from the Hadamard Code

We begin by describing a 2-query linear PCP based on the Hadamard PCP from [ALM+98, IKO07], extended to work over arbitrary finite fields $\mathbb{F}$. Our presentation follows that from [BCI+13, BISW17]. We begin by describing the 3-query variant from previous works [IKO07, BCI+13], and then show that a simple variant allows us to combine two of the queries, yielding a 2-query linear PCP.

Construction description. Let $C\colon \mathbb{F}_p^n \times \mathbb{F}_p^h \to \mathbb{F}_p^t$ be an arithmetic circuit of size s over a finite field $\mathbb{F}_p$. Let z_i denote the value of the i^{th} wire of C on an input $\mathbf{x} \in \mathbb{F}_p^n$ and a witness $\mathbf{w} \in \mathbb{F}_p^h$. The wires z_i for $i \in [n]$ correspond to the values of the input wires, and the wires $z_{s-i+1} = 0$ for $i \in [t]$ correspond to the values of the output wires. An honestly generated proof consists of a vector $\boldsymbol{\pi} \in \mathbb{F}_p^{s+s^2}$ whose components are the values $z_1, \ldots, z_s \in \mathbb{F}_p$ and the products $z_i z_j \in \mathbb{F}_p$ for all $i, j \in [s]$. If we define $\mathbf{z} = [z_1, \ldots, z_s]$, then we can write $\boldsymbol{\pi} = [\mathbf{z}, \mathbf{z} \otimes \mathbf{z}]$. To verify the proof, the verifier performs the following four consistency checks:

1. **Consistency of $\boldsymbol{\pi}$:** each entry $z_i z_j$ is the product of the corresponding z_i and z_j.
2. **Input consistency:** $z_i = x_i$ for all $i \in [n]$.
3. **Output consistency:** $z_{s-i+1} = 0$ for all $i \in [t]$.
4. **Gate consistency:** For each gate, the value of the output wire is consistent with the value of its input wires. In particular, for addition gates with input wires i, j and output wire k, this corresponds to checking that $z_i + z_j - z_k = 0$. For the multiplication gates with input wires i, j and output wire k, this corresponds to checking that $z_i z_j - z_k = 0$.

These consistency checks can be verified as follows:

- To verify the first condition, the verifier requests a random linear combination of the z_i's and a corresponding linear combination of the product terms $z_i z_j$'s whose sum corresponds to the square of the first query. Namely, the verifier samples $\mathbf{v} \xleftarrow{R} \mathbb{F}_p^s$ and constructs query vectors $\mathbf{q}_1 = [\mathbf{v}, 0^{s^2}]$ and $\mathbf{q}_2 = [0^s, \mathbf{v} \otimes \mathbf{v}]$. The verifier then checks the relation $(\mathbf{q}_1^\mathsf{T}\boldsymbol{\pi})^2 \stackrel{?}{=} \mathbf{q}_2^\mathsf{T}\boldsymbol{\pi}$. If $\boldsymbol{\pi}$ is not of the form $[\mathbf{z}, \mathbf{z} \otimes \mathbf{z}]$ for some $\mathbf{z} \in \mathbb{F}_p^s$, then by the Schwartz-Zippel lemma, $(\mathbf{q}_1^\mathsf{T}\boldsymbol{\pi})^2 = \mathbf{q}_2^\mathsf{T}\boldsymbol{\pi}$ with probability at most $2/p$.
- The remaining consistency checks can be expressed as a linear system in terms of the variables $\mathbf{z}$ and $\mathbf{z} \otimes \mathbf{z}$. Namely, we can define a matrix $\mathbf{A}_C \in \mathbb{F}_p^{s \times (s+s^2)}$ and a vector $\mathbf{b}_C \in \mathbb{F}_p^{s-n}$ where $\mathbf{A}_C$ and $\mathbf{b}_C$ are functions of C only (and which can be computed in linear time from C), such that $\mathbf{z}$ is a satisfying assignment to the wires of C only if

$$\mathbf{A}_C \begin{bmatrix} \mathbf{z} \\ \mathbf{z} \otimes \mathbf{z} \end{bmatrix} = \begin{bmatrix} \mathbf{x} \\ \mathbf{b}_C \end{bmatrix}.$$

  To check that this system is satisfied, the verifier samples $\mathbf{u} \xleftarrow{R} \mathbb{F}_p^s$ and checks the relation

$$\mathbf{u}^\mathsf{T}\mathbf{A}_C \begin{bmatrix} \mathbf{z} \\ \mathbf{z} \otimes \mathbf{z} \end{bmatrix} \stackrel{?}{=} \mathbf{u}^\mathsf{T} \begin{bmatrix} \mathbf{x} \\ \mathbf{b}_C \end{bmatrix} = \mathbf{u}_\mathbf{x}^\mathsf{T}\mathbf{x} + \mathbf{u}_\mathbf{b}^\mathsf{T}\mathbf{b}_C$$

  where $\mathbf{u}_\mathbf{x} \in \mathbb{F}_p^n$ denotes the first n components of $\mathbf{u}$ and $\mathbf{u}_\mathbf{b}$ denotes the other $s - n$ components. Concretely, the verifier sets $\mathbf{q}_3 \leftarrow \mathbf{u}^\mathsf{T}\mathbf{A}_C$ and checks the relation $\mathbf{q}_3^\mathsf{T}\boldsymbol{\pi} \stackrel{?}{=} \mathbf{u}_\mathbf{x}^\mathsf{T}\mathbf{x} + \mathbf{u}_\mathbf{b}^\mathsf{T}\mathbf{b}_C$. Assuming that $\boldsymbol{\pi}$ is consistent (i.e., of the form $[\mathbf{z}, \mathbf{z} \otimes \mathbf{z}]$ for some $\mathbf{z} \in \mathbb{F}_p^s$), then by the Schwartz-Zippel lemma, the verifier will reject a inconsistent wire assignment with probability at least $1/p$.

As described, the verifier prepares three queries $\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3$ and receives responses $a_1 = \mathbf{q}_1^\mathsf{T}\boldsymbol{\pi}$, $a_2 = \mathbf{q}_2^\mathsf{T}\boldsymbol{\pi}$, and $a_3 = \mathbf{q}_3^\mathsf{T}\boldsymbol{\pi}$ and checks that $a_1^2 \stackrel{?}{=} a_2$ and $a_3 \stackrel{?}{=} \mathbf{u}_\mathbf{x}^\mathsf{T}\mathbf{x} + \mathbf{u}_\mathbf{b}^\mathsf{T}\mathbf{b}_C$. We make several observations:

- The decision procedure is fully linear and only depends on $\mathbf{u}_\mathbf{x}^\mathsf{T}, \mathbf{u}_\mathbf{b}^\mathsf{T}\mathbf{b}_C$.
- The responses a_2 and a_3 are used in a linear manner at the decision procedure, and depend on *independently-sampled* randomness. Thus, instead of checking them independently, we can instead check the joint relation

$$a_1^2 - a_2 + (a_3 - \mathbf{u}_\mathbf{x}^\mathsf{T}\mathbf{x} - \mathbf{u}_\mathbf{b}^\mathsf{T}\mathbf{b}_C) \stackrel{?}{=} 0. \tag{B.1}$$

  By the Schwartz-Zippel lemma, if either of the two verification relations does not hold, then Eq. (B.1) is nonzero with probability at least $2/p$. The key is that we can check Eq. (B.1) using just two queries by setting $\bar{\mathbf{q}}_1 = \mathbf{q}_1$ and $\bar{\mathbf{q}}_2 = \mathbf{q}_3 - \mathbf{q}_2$ and testing $\bar{a}_1^2 + \bar{a}_2 - \mathbf{u}_\mathbf{x}^\mathsf{T}\mathbf{x} - \mathbf{u}_\mathbf{b}^\mathsf{T}\mathbf{b}_C \stackrel{?}{=} 0$, where $\bar{a}_1 = \bar{\mathbf{q}}_1^\mathsf{T}\boldsymbol{\pi}$ and $\bar{a}_2 = \bar{\mathbf{q}}_2^\mathsf{T}\boldsymbol{\pi}$. This precisely corresponds to Eq. (B.1) above.

We give the formal construction below:

Construction B.1 (2-Query Linear PCP from the Walsh-Hadamard Code)**.** Let $C \colon \mathbb{F}_p^n \times \mathbb{F}_p^h \to \mathbb{F}_p^t$ be an arithmetic circuit of size s over $\mathbb{F}_p$. We construct a 2-query linear PCP $\Pi_{\mathsf{LPCP}} = (\mathcal{Q}_{\mathsf{LPCP}}, \mathcal{P}_{\mathsf{LPCP}}, \mathcal{D}_{\mathsf{LPCP}})$ for $\mathcal{R}_C$ as follows:

- $\mathcal{Q}_{\mathsf{LPCP}}$: Let $\mathbf{A}_C \in \mathbb{F}_p^{s \times (s+s^2)}$ and $\mathbf{b}_C \in \mathbb{F}_p^{s-n}$ be the circuit-specific linear system that correspond to the circuit consistency checks. The query algorithm then computes two query vectors as follows $(\mathbf{q}_1, \mathbf{q}_2)$:
  1. Sample $\mathbf{v} \xleftarrow{R} \mathbb{F}_p^s$. Set $\mathbf{q}_1^\mathsf{T} = [\mathbf{v}^\mathsf{T}, 0^{s^2}]$.
  2. Sample $\mathbf{u} \xleftarrow{R} \mathbb{F}_p^s$. Let $\mathbf{q}_2^\mathsf{T} = \mathbf{u}^\mathsf{T}\mathbf{A}_C - [0^s, (\mathbf{v} \otimes \mathbf{v})^\mathsf{T}]$, and $\mathbf{u}_\mathbf{x} \in \mathbb{F}_p^n$ be the first n components of $\mathbf{u}$, and $\mathbf{u}_\mathbf{b} \in \mathbb{F}_p^{s-n}$ be the rest of the components of $\mathbf{u}$.

The query sets $\mathbf{q}_{\mathsf{inp}} = \mathbf{u}_{\mathbf{x}}$ and $\mathbf{Q} \in \mathbb{F}_p^{(s+s^2)\times 2}$ to be the matrix whose columns are $\mathbf{q}_1$ and $\mathbf{q}_2$. It also computes $u_C \leftarrow \mathbf{u}_{\mathbf{b}}^{\mathsf{T}}\mathbf{b}_C$ and sets $\mathsf{st} = u_C$. Finally, it outputs st, $\mathbf{q}_{\mathsf{inp}}$, and $\mathbf{Q}$.

- $\mathcal{P}_{\mathsf{LPCP}}(\mathbf{x}, \mathbf{w})$: On input $(\mathbf{x}, \mathbf{w}) \in \mathbb{F}_p^n \times \mathbb{F}_p^h$ where $C(\mathbf{x}, \mathbf{w}) = \mathbf{0}$, the prover computes the values $z_1, \ldots, z_s$ for the wires of C. It sets $\mathbf{z} = [z_1, \ldots, z_s] \in \mathbb{F}_p^s$ and outputs the proof $\boldsymbol{\pi} = [\mathbf{z}, \mathbf{z} \otimes \mathbf{z}]^{\mathsf{T}} \in \mathbb{F}_p^{s+s^2}$.
- $\mathcal{D}_{\mathsf{LPCP}}(\mathsf{st}, a_{\mathsf{inp}}, \mathbf{a})$ On input the state $\mathsf{st} = u_C$, an input-dependent response a_{inp}, and a vector of responses $\mathbf{a} = (a_1, a_2) \in \mathbb{F}_p^2$, the verifier outputs 1 if $a_1^2 + a_2 = a_{\mathsf{inp}} + u_C$, and 0 otherwise.

Theorem B.2 (Completeness)**.** *Construction B.1 satisfies perfect completeness.*

Proof. Take any $\mathbf{x}, \mathbf{w}$ where $C(\mathbf{x}, \mathbf{w}) = 0^t$. Let $(\mathsf{st}, \mathbf{q}_{\mathsf{inp}}, \mathbf{Q}) \leftarrow \mathcal{Q}_{\mathsf{LPCP}}$ and $\boldsymbol{\pi} \leftarrow \mathcal{P}_{\mathsf{LPCP}}(\mathbf{x}, \mathbf{w})$. By construction, $\boldsymbol{\pi} = [\mathbf{z}, \mathbf{z} \otimes \mathbf{z}]$, where $\mathbf{z}$ is the wire values corresponding to $C(\mathbf{x}, \mathbf{w})$. Let $a_1 \leftarrow \mathbf{q}_1^{\mathsf{T}}\boldsymbol{\pi}$ and $a_2 \leftarrow \mathbf{q}_2^{\mathsf{T}}\boldsymbol{\pi}$, where $\mathbf{q}_1$ and $\mathbf{q}_2$ are the columns of $\mathbf{Q}$. Then,

$$a_1^2 + a_2 = (\mathbf{v}^{\mathsf{T}}\mathbf{z})^2 + \mathbf{u}^{\mathsf{T}}\mathbf{A}_C\boldsymbol{\pi} - (\mathbf{v} \otimes \mathbf{v})^{\mathsf{T}}(\mathbf{z} \otimes \mathbf{z}) = (\mathbf{v}^{\mathsf{T}}\mathbf{z})^2 + \mathbf{u}^{\mathsf{T}}\mathbf{A}_C\boldsymbol{\pi} - (\mathbf{v}^{\mathsf{T}}\mathbf{z})(\mathbf{v}^{\mathsf{T}}\mathbf{z}) = \mathbf{u}^{\mathsf{T}}\mathbf{A}_C\boldsymbol{\pi}.$$

By construction of $\mathbf{A}_C$, if $\boldsymbol{\pi} = [\mathbf{z}, \mathbf{z} \otimes \mathbf{z}]^{\mathsf{T}}$ for a valid wire labeling, then

$$\mathbf{u}^{\mathsf{T}}\mathbf{A}_C\boldsymbol{\pi} = \mathbf{u}_{\mathbf{x}}^{\mathsf{T}}\mathbf{x} + \mathbf{u}_{\mathbf{b}}^{\mathsf{T}}\mathbf{b}_C = a_{\mathsf{inp}} + u_C. \qquad \square$$

Theorem B.3 (Strong Soundness)**.** *Construction B.1 has strong soundness error $2/p$.*

Proof. Take any statement $\mathbf{x} \notin \mathcal{L}$ and consider any affine proof strategy $\boldsymbol{\pi}^* = [\boldsymbol{\pi}_1^*, \boldsymbol{\pi}_2^*] \in \mathbb{F}_p^{s+s^2}$, where $\boldsymbol{\pi}_1^* \in \mathbb{F}_p^s$ and $\boldsymbol{\pi}_2^* \in \mathbb{F}_p^{s^2}$ and $\delta_1^*, \delta_2^* \in \mathbb{F}_p$. Let $(\mathsf{st}, \mathbf{q}_{\mathsf{inp}}, \mathbf{Q}) \leftarrow \mathcal{Q}_{\mathsf{LPCP}}$, where $\mathsf{st} = u_C$ and $\mathbf{q}_{\mathsf{inp}} = \mathbf{u}_{\mathbf{x}}$. Let $\mathbf{q}_1, \mathbf{q}_2$ be the columns of $\mathbf{Q}$. Write $\mathbf{q}_1^{\mathsf{T}} = [\mathbf{v}^{\mathsf{T}}, 0^{s^2}]$ and $\mathbf{q}_2^{\mathsf{T}} = \mathbf{u}^{\mathsf{T}}\mathbf{A}_C - [0^s, (\mathbf{v} \otimes \mathbf{v})^{\mathsf{T}}]$. Consider $a_1 \leftarrow \mathbf{q}_1^{\mathsf{T}}\boldsymbol{\pi} + \delta_1$ and $a_2 \leftarrow \mathbf{q}_2^{\mathsf{T}}\boldsymbol{\pi} + \delta_2$. The verifier accepts if

$$a_1^2 + a_2 - a_{\mathsf{inp}} - u_C = (\mathbf{v}^{\mathsf{T}}\boldsymbol{\pi}_1^* + \delta_1^*)^2 + \mathbf{u}^{\mathsf{T}}\mathbf{A}_C\boldsymbol{\pi}^* - (\mathbf{v} \otimes \mathbf{v})^{\mathsf{T}}\boldsymbol{\pi}_2^* + \delta_2^* - \mathbf{u}_{\mathbf{x}}^{\mathsf{T}}\mathbf{x} - \mathbf{u}_{\mathbf{b}}^{\mathsf{T}}\mathbf{b}_C = 0. \tag{B.2}$$

We consider two possibilities:

- Suppose that $\boldsymbol{\pi}_2^* \neq \boldsymbol{\pi}_1^* \otimes \boldsymbol{\pi}_1^*$. Then Eq. (B.2) is not identically zero in the variables $\mathbf{v}$.
- Suppose that $\boldsymbol{\pi}^* = [\mathbf{z}^*, \mathbf{z}^* \otimes \mathbf{z}^*]$ for some $\mathbf{z}^* \in \mathbb{F}_p^s$. Since $\mathbf{x} \notin \mathcal{L}$, there is no consistent labeling of the wires of C with input $\mathbf{x}$ and output values 0^t (i.e., no value $\mathbf{z}^*$ can satisfy the input, output, and circuit consistency constraints). This means that $\mathbf{A}_C\boldsymbol{\pi}^* \neq [\mathbf{x}, \mathbf{b}_C]^{\mathsf{T}}$, and so $\mathbf{u}^{\mathsf{T}}\mathbf{A}_C\boldsymbol{\pi}^* - \mathbf{u}_{\mathbf{x}}^{\mathsf{T}}\mathbf{x} - \mathbf{u}_b^{\mathsf{T}}\mathbf{b}_C$ is not the identically zero polynomial in the variables $\mathbf{u}$. Thus, in this case, Eq. (B.2) is not identically zero in the variables $\mathbf{u}$.

In both cases, the verification relation Eq. (B.2) is *not* identically zero in either $\mathbf{u}$ or $\mathbf{v}$. Since these variables are sampled uniformly from $\mathbb{F}_p$, and Eq. (B.2) is a polynomial of total degree 2, we appeal to Schwartz-Zippel to conclude that over the verifier's choice of randomness, Eq. (B.2) holds with probability at most $2/p$, and soundness follows. We can apply the same analysis as above to argue strong soundness (with the same soundness error). $\square$

Remark B.4 (Reducing Query Length)**.** We note that in the Hadamard LPCP from Construction B.1, the proof $\boldsymbol{\pi}$ has the form $\boldsymbol{\pi} = [\mathbf{z}, \mathbf{z} \otimes \mathbf{z}]$. By default, $\mathbf{z} \otimes \mathbf{z}$ contains both the value of $z_i z_j$ as well as the value of $z_j z_i$ whenever $i \neq j$. We can reduce the query size slightly by only including *one* copy of each pairwise product. This allows us to reduce the query length from $s + s^2$ to $s + s + s(s-1)/2 = (s^2 + 3s)/2$.

Remark B.5 (Honest-Verifier Zero Knowledge)**.** As noted in [BCI+13, Remark A.4], it is straightforward to obtain a perfect honest-verifier zero-knowledge linear PCP from the Hadamard PCP by introducing a dummy wire to the circuit and having the prover assigning the wire a *random* value. The verifier then performs all of the checks with respect to this new circuit. We refer to [BCI+13, Remark A.4] for more details.

Remark B.6 (Knowledge Soundness)**.** We further note that Construction B.1 satisfies knowledge soundness (Definition 2.7). From Theorem B.3 it is clear that any linear strategy $\boldsymbol{\pi}^*, \boldsymbol{\delta}^*$ that convinces $\mathcal{D}_{\mathsf{LPCP}}$ with probability greater than ε must encode a satisfying assignment to the circuit C (with $\boldsymbol{\delta}^* = \mathbf{0}$). Therefore, the extractor $\mathcal{E}$ can simply query $\boldsymbol{\pi}^*$ with $|\mathbf{w}|$ unit vectors to recover the witness $\mathbf{w}$.

Corollary B.7 (2-Query Hadamard Linear PCP)**.** *Let $C\colon \mathbb{F}_p^n \times \mathbb{F}_p^h \to \mathbb{F}_p^t$ be an arithmetic circuit of size s over $\mathbb{F}_p$. Then there exists a 2-query, degree-2 linear PCP with perfect zero-knowledge for the relation $\mathcal{R}_C$ with query length $t = (s^2 + 3s)/2 = O(s^2)$ and strong soundness error $2/p$.*

B.2 Bounded Linear PCP from the Hadamard Code

It is straightforward to adapt the Hadamard linear PCP from Construction B.1 to obtain a bounded linear PCP. Here, we will focus on the particular setting of Boolean circuits, so the input and output wires of the circuit are *binary*. Our construction naturally generalizes to general circuits over the integers with bounded wire values (provided that each gate in the circuit computes a quadratic function of its inputs). Below, we present our construction for Boolean circuits:

Construction B.8 (2-Query Bounded Linear PCP)**.** Let $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ be a Boolean circuit of size s. Let $b_1, b_2\colon \mathbb{N} \to \mathbb{N}$ be bound functions. We construct a 2-query bounded linear PCP $\Pi_{\mathsf{LPCP}} = (\mathcal{Q}_{\mathsf{LPCP}}, \mathcal{P}_{\mathsf{LPCP}}, \mathcal{D}_{\mathsf{LPCP}})$ over $\mathbb{F}_p$ for the relation $\mathcal{R}_C$ as in Construction B.1, but with the following differences:

- **Circuit consistency checks:** In Construction B.1, we defined circuit consistency checks for arithmetic circuits over $\mathbb{F}_p$ where the gates were addition and multiplication gates over $\mathbb{F}_p$ and the wire values were arbitrary $\mathbb{F}_p$ elements. Here, we work with Boolean circuits so that in the honest execution, all of the wire values are binary valued. In Table 3, we describe how to implement linear consistency checks for several standard Boolean gates. We assume that each gates has exactly two input wires. Using the same procedure as Construction B.1, we then define the circuit consistency check $\mathbf{A}_C \in \mathbb{F}_p^{s \times (s+s^2)}$ and $\mathbf{b}_C \in \mathbb{F}_p^{s-n}$ for a Boolean circuit C. Note that we include the same input-consistency and output-consistency checks as before.
- **Bounded query domain:** The query-generation algorithm $\mathcal{Q}_{\mathsf{LPCP}}$ now takes an additional bound parameter $\tau \in \mathbb{N}$ and instead of sampling $\mathbf{u}, \mathbf{v}$ from the full domain, it instead samples $\mathbf{u}, \mathbf{v} \xleftarrow{R} [-\tau/2, \tau/2]^s$. The verification state st also includes the bound parameter τ.
- **Bound checking at verification time:** On input $\mathsf{st} = (u_C, \tau)$, an input-dependent response a_{inp}, and a vector of responses $\mathbf{a} = (a_1, a_2) \in \mathbb{F}_p^2$, the verifier additionally checks that $a_1 \in [-b_1(\tau), b_1(\tau)]$ and $a_2 \in [-b_2(\tau), b_2(\tau)]$.

Everything else is constructed as in Construction B.1.

Theorem B.9 (Completeness)**.** *Construction B.8 satisfies perfect completeness.*

Proof. Same as the proof of Theorem B.2. □

Theorem B.10 (Soundness)**.** *Construction B.8 has strong soundness error $2/\tau$.*

Proof. Follows from the same argument as in the proof of Theorem B.3. The only difference is that $\mathbf{u}, \mathbf{v}$ are now sampled uniformly from an interval of size τ rather than $|\mathbb{F}_p|$. □

Remark B.11 (Knowledge Soundness)**.** Construction B.8 has $2/\tau$ knowledge soundness (Definition 2.7). The argument is identical to the argument in Remark B.6

Theorem B.12 (Bounded)**.** *Construction B.8 is a bounded linear PCP where $b_1(\tau) = s\tau/2$ and $b_2(\tau) = 2(b_1(\tau))^2$. Moreover, for any $\varepsilon > 0$, Construction B.8 is bounded with respect to bound functions $b'_1(\tau) = \tau\sqrt{s/2 \cdot \ln(2/\varepsilon)}$ and $b'_2(\tau) = 2(b'_1(\tau))^2$ with probability at least $1 - \varepsilon$.*

Gate Type	Input Wires	Output Wire	Linear Checks
AND	z_i, z_j	z_k	$z_k - z_{ij} = 0$
NOT	z_i	z_k	$z_i + z_k = 1$
OR	z_i, z_j	z_k	$z_k - z_i - z_j + z_{ij} = 0$
XOR	z_i, z_j	z_k	$z_k - z_i - z_j + 2z_{ij} = 0$
NAND	z_i, z_j	z_k	$z_k + z_{ij} = 1$

Table 3: Linear checks for common Boolean gates. We assume that each gate has two input wires and a single output wire. It is straightforward to extend this approach to arbitrary Boolean gates whose output value can be expressed as a quadratic function of its input values. Recall that the Hadamard linear PCP consists of a vector of wire labels $\mathbf{z}$ together with their pairwise products $\mathbf{z} \otimes \mathbf{z}$. We write z_i to denote the i^{th} wire label in $\mathbf{z}$ and z_{ij} to denote the entry in $\mathbf{z} \otimes \mathbf{z}$ that corresponds to the product $z_i z_j$.

Proof. Take any $x \in \{0,1\}^n$ and $w \in \{0,1\}^h$ where $C(x,w) = 1$. Let $(\mathsf{st}, \mathbf{q}_{\mathsf{inp}}, \mathbf{Q}) \xleftarrow{R} \mathcal{Q}_{\mathsf{LPCP}}(\tau)$ and $\boldsymbol{\pi} \leftarrow \mathcal{P}_{\mathsf{LPCP}}(x, w)$ where $\mathsf{st} = (u_C, \tau)$. By construction, $\boldsymbol{\pi} = [\mathbf{z}, \mathbf{z} \otimes \mathbf{z}]$, where $\mathbf{z} \in \mathbb{F}_p^s$ are the wire labels of circuit C on input (x, w). Since C is a Boolean circuit, this means that every value in $\boldsymbol{\pi}$ is binary-valued. Next, let $\mathbf{q}_1, \mathbf{q}_2$ denote the two columns of $\mathbf{Q}$. By construction $\mathbf{q}_1$ contains exactly s nonzero entries, each of which is drawn from the interval $[-\tau/2, \tau/2]$. Thus, $\mathbf{q}_1^{\mathsf{T}}\boldsymbol{\pi} \in [-s\tau/2, s\tau/2]$. By completeness, we know that the verification relation is satisfied, in which case

$$\mathbf{q}_2^{\mathsf{T}}\boldsymbol{\pi} = \mathbf{q}_{\mathsf{inp}}^{\mathsf{T}}\mathbf{x} + u_C - (\mathbf{q}_1^{\mathsf{T}}\boldsymbol{\pi})^2.$$

Here $\mathbf{q}_{\mathsf{inp}}^{\mathsf{T}}\mathbf{x} = \mathbf{u}_{\mathbf{x}}^{\mathsf{T}}\mathbf{x}$ and $u_C = \mathbf{u}_{\mathbf{b}}^{\mathsf{T}} b_C$, where each component of $\mathbf{u}_{\mathbf{x}}$ and $\mathbf{u}_{\mathbf{b}}$ are sampled from the interval $[-\tau/2, \tau/2]$. For the Boolean gates in Table 3, we have that each component of $\mathbf{b}_C$ is binary-valued, as is the input $\mathbf{x}$. Thus, if we work over the integers, then we can bound

$$|\mathbf{q}_2^{\mathsf{T}}\boldsymbol{\pi}| \leq |\mathbf{q}_{\mathsf{inp}}^{\mathsf{T}}\mathbf{x}| + |u_C| + |\mathbf{q}_1^{\mathsf{T}}\boldsymbol{\pi}|^2 \leq s\tau + (b_1(\tau))^2 = 2b_1(\tau) + (b_1(\tau))^2 \leq 2(b_1(\tau))^2.$$

Thus, Construction B.8 is bounded with respect to b_1, b_2. For the second part of the claim, it suffices to bound the value of $\mathbf{q}_1^{\mathsf{T}}\boldsymbol{\pi}$. This follows by a straightforward application of Hoeffding's inequality. Namely, for any bound $B > 0$, and working over the integers, we have

$$\Pr[|\mathbf{q}_1^{\mathsf{T}}\boldsymbol{\pi}| \geq B] \leq 2\exp\left(-\frac{2B^2}{\mathsf{wt}(\mathbf{q}_1) \cdot \tau^2}\right) \leq 2\exp\left(-\frac{2B^2}{s\tau^2}\right).$$

Substituting $B = b_1'(\tau)$, we have

$$\Pr[|\mathbf{q}_1^{\mathsf{T}}\boldsymbol{\pi}| \geq b_1'(\tau)] \leq 2\exp\left(-\frac{2\tau^2 s/2 \cdot \ln(2/\varepsilon)}{s\tau^2}\right) = \varepsilon.$$

The claim then follows by a similar argument as before. □

B.3 Zero-Knowledge via Noise Smudging

Our bounded linear PCP from Construction B.8 does not provide zero-knowledge (even against honest verifiers). Using the same idea from [BCI+13] (see Remark B.5), we can adapt Construction B.8 to satisfy δ-HVZK while remaining a bounded linear PCP. Namely, as in [BCI+13], we introduce a dummy wire to the circuit, and the prover assigns a random value (chosen from a small interval) when constructing the proof. For completeness, we provide the full construction below:

Construction B.13 (2-Query Bounded Linear PCP with Zero-Knowledge)**.** Fix a parameter $\delta > 0$. Let b_1, b_2 be bound functions (that depend on both the bound parameter τ and the zero-knowledge parameter δ). Let $C\colon \{0,1\}^n \times \{0,1\}^h \to \{0,1\}$ be a Boolean circuit of size s. We construct a 2-query bounded linear PCP $\Pi_{\mathsf{LPCP}} = (\mathcal{Q}_{\mathsf{LPCP}}, \mathcal{P}_{\mathsf{LPCP}}, \mathcal{D}_{\mathsf{LPCP}})$ over a finite field $\mathbb{F}_p$ as follows:

- $\mathcal{Q}_{\mathsf{LPCP}}(\tau)$: On input the bound parameter τ, the query-generation algorithm samples $\mathbf{u}, \mathbf{v}' \xleftarrow{R} [-\tau/2, \tau/2]^s$, and defines $\mathbf{v} = [(\mathbf{v}')^\mathsf{T}, 1]^\mathsf{T} \in \mathbb{F}_p^{s+1}$. Let $\mathbf{u_x} \in \mathbb{F}_p^n$ be the first n components of $\mathbf{u}$ and $\mathbf{u_b} \in \mathbb{F}_p^{s-n}$ be the remaining $s-n$ components of $\mathbf{u}$. The query algorithm constructs the circuit consistency check matrix $\mathbf{A}_C \in \mathbb{F}_p^{s \times ((s+1)+(s+1)^2)}$ and vector $\mathbf{b}_C \in \mathbb{F}_p^{s-n}$ as in Construction B.8. It defines the queries $\mathbf{q}_1^\mathsf{T} = [\mathbf{v}^\mathsf{T}, 0^{(s+1)^2}] \in \mathbb{F}_p^{(s+1)+(s+1)^2}$ and $\mathbf{q}_2^\mathsf{T} = \mathbf{u}^\mathsf{T}\mathbf{A}_C - [0^{s+1}, (\mathbf{v} \otimes \mathbf{v})^\mathsf{T}] \in \mathbb{F}_p^{(s+1)+(s+1)^2}$. Let $\mathbf{Q} \in \mathbb{F}_p^{((s+1)+(s+1)^2) \times 2}$ be the matrix whose columns are $\mathbf{q}_1$ and $\mathbf{q}_2$. It computes $u_C \leftarrow \mathbf{u_b}^\mathsf{T}\mathbf{b}_C$, and outputs st, $\mathbf{q}_{\mathsf{inp}} = \mathbf{u_x}$, and $\mathbf{Q}$.
- $\mathcal{P}_{\mathsf{LPCP}}(\tau, x, w)$: On input $(x, w) \in \{0,1\}^n \times \{0,1\}^h$, the prover evaluates $C(x, w)$ and sets $z_1, \ldots, z_s$ to be the wire values for C. Let $B = 2\tau\sqrt{s/2 \cdot \ln(4/\delta)}/\delta$. The prover samples a "smudging factor" $z_{s+1} \xleftarrow{R} [-B, B]$. Let $\mathbf{z} \in \mathbb{F}_p^{s+1}$ be the vector whose elements are $z_1, \ldots, z_s, z_{s+1}$. The prover outputs the proof $\boldsymbol{\pi} = [\mathbf{z}, \mathbf{z} \otimes \mathbf{z}] \in \mathbb{F}_p^{(s+1)+(s+1)^2}$.
- $\mathcal{D}_{\mathsf{LPCP}}(\mathsf{st}, a_{\mathsf{inp}}, \mathbf{a})$: On input the verification state $\mathsf{st} = (\tau, u_C)$, an input-dependent response a_{inp}, and responses $\mathbf{a} = (a_1, a_2) \in \mathbb{F}_p^2$, the verifier first checks that $a_1 \in [-b_1(\tau, \delta), b_1(\tau, \delta)]$ and $a_2 \in [-b_2(\tau, \delta), b_2(\tau, \delta)]$. If both checks pass, the verifier outputs 1 if $a_1^2 + a_2 = a_{\mathsf{inp}} + u_C$, and 0 otherwise.

Theorem B.14 (Completeness)**.** *Construction B.13 satisfies perfect completeness.*

Proof. Same as the proofs of Theorems B.2 and B.9. □

Theorem B.15 (Soundness)**.** *Construction B.13 has soundness error* $2/\tau$*.*

Proof. Same as the proof of Theorem B.10. □

Remark B.16. Construction B.13 has knowledge soundness $2/\tau$. The argument is identical to the argument at Remark B.6

Remark B.17 (Strong Soundness of Construction B.13)**.** Construction B.13 does *not* provide strong soundness. Notably, a malicious prover can choose the smudging factor z_{s+1} to be outside the allowable range. By design, z_{s+1} adds a linear shift to the first response a_1. Since the verifier checks whether $a_1 \in [-b_1(\tau, \delta), b_1(\tau, \delta)]$, there exist values of z_{s+1} near the "border" that will cause the honest verifier to reject with probability between 1 and $2/\tau$, which violates strong soundness.

Theorem B.18 (Bounded)**.** *Construction B.13 is a bounded linear PCP with bound functions* $b_1(\tau, \delta) = s\tau/2 + 2\tau\sqrt{s/2 \cdot \ln(4/\delta)}/\delta$*, and* $b_2(\tau) = 2(b_1(\tau))^2$*. Moreover, for any* $\varepsilon > 0$*, Construction B.13 is bounded with respect to bound functions* $b_1'(\tau) = \tau\sqrt{s/2}\big(\sqrt{\ln(2/\varepsilon)} + 2/\delta\sqrt{\ln(4/\delta)}\big)$ *and* $b_2'(\tau) = 2(b_1(\tau))^2$ *with probability at least* $1 - \varepsilon$*.*

Proof. Same as the proof of Theorem B.12, except the value of a_1 can now increase by the magnitude of the smudging factor z_{s+1}, which is sampled from a bounded interval with magnitude at most $2\tau\sqrt{s/2 \cdot \ln(4/\delta)}/\delta$. The claim follows. □

Theorem B.19 (δ-HVZK)**.** *Construction B.13 satisfies* δ-HVZK*.*

Proof. Our analysis relies on the following standard noise smudging lemma.

Lemma B.20 (Noise Smudging (cf. [AJL+12]))**.** *Let* B_1 *and* B_2 *be positive integers. Let* $x \in [-B_1, B_1]$ *be a fixed integer. Sample* $y \xleftarrow{R} [-B_2, B_2]$*. Then, the distribution of* y *is* ε*-close to the distribution of* $y + x$ *for* $\varepsilon = B_1/B_2$*.*

We construct a simulator $\mathcal{S}$ as follows:

- On input a bound parameter $\tau \in \mathbb{N}$ and a statement $\mathbf{x} \in \mathcal{L}$, sample $(\mathsf{st}, \mathbf{q}_{\mathsf{inp}}, \mathbf{Q}) \leftarrow \mathcal{Q}_{\mathsf{LPCP}}(\tau)$, where $\mathsf{st} = u_C$. Let $a_{\mathsf{inp}} \leftarrow \mathbf{q}_{\mathsf{inp}}^\mathsf{T}\mathbf{x}$.

- Sample $\tilde{a}_1 \xleftarrow{R} [-B, B]$ where $B = 2\tau\sqrt{s/2 \cdot \ln(4/\delta)}/\delta$ and set $\tilde{a}_2 \leftarrow a_{\mathsf{inp}} + u_C - a_1^2$. Let $\tilde{\mathbf{a}} = (\tilde{a}_1, \tilde{a}_2) \in \mathbb{F}_p^2$.
- Output $(\mathsf{st}, \mathbf{q}_{\mathsf{inp}}, \mathbf{Q}, a_{\mathsf{inp}}, \tilde{\mathbf{a}})$.

By construction, the simulator samples $\mathsf{st}, \mathbf{q}_{\mathsf{inp}}, \mathbf{Q}, a_{\mathsf{inp}}$ from exactly the same distribution as in the real distribution. It suffices to argue that the simulated set of responses $\tilde{\mathbf{a}}$ is properly distributed. In the real distribution, $\mathbf{a} \leftarrow \mathbf{Q}^\mathsf{T}\boldsymbol{\pi}$ where $\boldsymbol{\pi} \leftarrow \mathcal{P}_{\mathsf{LPCP}}(\tau, x, w)$. First, since $x \in \mathcal{L}$, in the real distribution, the verifier accepts which means that $a_1^2 + a_2 = a_{\mathsf{inp}} + u_C$. This means that as long as a_1 and $\tilde{a}_1$ are δ-close, then the real and simulated distributions are δ-close. We consider the distribution of a_1 in the real distribution. In the real distribution

$$a_1 = \mathbf{q}_1^\mathsf{T}\boldsymbol{\pi} = (\mathbf{v}')^\mathsf{T}\mathbf{z}' + z_{s+1},$$

where $\mathbf{z}' \in \mathbb{F}_p^s$ denotes the first s components of $\mathbf{z}$. By the same analysis as in the proof of Theorem B.12, we have that for $B' = \tau\sqrt{s/2 \cdot \ln(4/\delta)}$,

$$\Pr[|(\mathbf{v}')^\mathsf{T}\mathbf{z}'| \geq B'] \leq \delta/2.$$

Thus, with probability $1 - \delta/2$, $(\mathbf{v}')^\mathsf{T}\mathbf{z}' \in [-B', B']$. Since $B'/B = \delta/2$, we can apply Lemma B.20 to conclude that over the randomness of z_{s+1}, the distribution of $a_1 = (\mathbf{v}')^\mathsf{T}\mathbf{z}' + z_{s+1}$ in the real distribution is $(\delta/2)$-close to the distribution of $\tilde{a}_1$ in the simulated distribution (where a_1 is uniform over $[-\delta', \delta']$). □

C The Generic Group Model

In this work, we analyze the security of some of our constructions in the generic group model [Nec94, Sho97]. In the generic group model, access to the group elements is replaced by "handles." An adversary in the generic group model is also given access to a stateful oracle which implements the group operation. The generic group oracle maintains internally a mapping from handles to group elements, which it uses in order to consistently answer the oracle queries. Thus, when a scheme is shown to satisfy some security property in the generic group model, it means that no efficient adversary that only applies the group operations as a black-box can break that security property. Our presentation is adapted from similar definitions from [Zim15, KLM+18].

Definition C.1 (Generic Group Oracle)**.** A generic group oracle is a stateful oracle $\mathcal{G}$ that responds to queries $\mathsf{GGM.Setup}, \mathsf{GGM.Encode}, \mathsf{GGM.Add}, \mathsf{GGM.Test}$ as follows:

- On a query $\mathsf{GGM.Setup}(1^\lambda)$, the generic group oracle samples two fresh nonces $\mathsf{pp}, \mathsf{sk} \xleftarrow{R} \{0,1\}^\lambda$ and a prime p. It outputs $(\mathsf{pp}, \mathsf{sk}, p)$. The oracle stores the values generated, initialize an empty table $\mathsf{T} \leftarrow \{\}$, and set the internal state so subsequent invocations of $\mathsf{GGM.Setup}$ fail (with output $\perp$).
- On a query $\mathsf{GGM.Encode}(k, x)$ where $k \in \{0,1\}^\lambda$, $x \in \mathbb{F}_p$, the oracle checks that $k = \mathsf{sk}$ (returning $\perp$ if the check fails). The oracle then generates a fresh nonce $\xi \xleftarrow{R} \{0,1\}^\lambda$, adds the entry $\xi \mapsto x$ to the table T, and replies with ξ.
- On a query $\mathsf{GGM.Add}(k, \xi_1, \xi_2)$ where $k, \xi_1, \xi_2 \in \{0,1\}^\lambda$, the oracle checks that $k = \mathsf{pp}$, that the handles ξ_1, ξ_2 are present in its internal table T, and are mapped to values $x_1, x_2 \in \mathbb{F}_p$, respectively (returning $\perp$ otherwise). If the checks pass, the oracle samples a fresh handle $\xi \xleftarrow{R} \{0,1\}^\lambda$ and adds the entry $\xi \mapsto (x_1 + x_2)$ to T, and replies with ξ. The addition oracle can be used to implement *scalar multiplication* by arbitrary $\mathbb{F}_p$ elements via repeated doubling.
- On a query $\mathsf{GGM.Test}(k, \xi)$ where $k, x \in \{0,1\}^\lambda$, the oracle checks that $k = \mathsf{pp}$, that the handle ξ is present in T, and that ξ maps to some value $x \in \mathbb{F}_p$ (returning $\perp$ otherwise). If the checks pass, the oracle returns "zero" if $x = 0 \in \mathbb{F}_p$ and "nonzero" otherwise.

Remark C.2 (Unique Encodings)**.** Many formulations, including [Sho97], model the generic group using a random injective function $\sigma\colon \mathbb{F}_p \to \{0,1\}^\lambda$. In this formulation, every value in $\mathbb{F}_p$ has a unique encoding, and there is no need for an explicit $\mathsf{GGM.Test}$ procedure ($\mathsf{GGM.Test}$ would just correspond to equality of bitstrings). Our formulation in Definition C.1 (which follows [Zim15, KLM+18]) samples a new *encoding* on every query and provides an explicit $\mathsf{GGM.Test}$ procedure for checking whether an element is an encoding of 0 (or equivalently, whether two elements are equal). We can implement a generic group model with unique encodings by using the $\mathsf{GGM.Test}$ procedure to test equality against the existing entries in the table T after each $\mathsf{GGM.Encode}$ and $\mathsf{GGM.Add}$ query, and returning the previously-computed handle if it is already present in the table. Otherwise, a new handle is sampled as usual. This transformation incurs a quadratic overhead in the number of queries. Thus, without loss of generality, we can assume fresh handles are output by $\mathsf{GGM.Encode}$ and $\mathsf{GGM.Add}$, and equality-checking is handled through an explicit algorithm $\mathsf{GGM.Test}$.

Remark C.3 (Oracle Queries as Formal Polynomials [Zim15, Remark 2.11, adapted])**.** Although the generic group oracle is defined formally in terms of "handles" (Definition C.1), it is oftentimes more conducive to regard each oracle query as referring to a formal query polynomial. The formal variables in this formal query polynomial are specified by the expressions supplied to the $\mathsf{GGM.Encode}$ oracle (as determined by the details of the construction), and the adversary can construct terms that refer to new polynomials by making queries to the group operation oracle $\mathsf{GGM.Add}$. Rather than operating on a "handle," each valid $\mathsf{GGM.Test}$ query refers to a formal query polynomial, and the result of the query is "zero" if the polynomial evaluates to zero when its variables are instantiated with the joint distribution over their values in $\mathbb{F}_p$ as generated in the real security game.

C.1 Linear Targeted Malleability of ElGamal

In this section, we show that the ElGamal encryption scheme [ElG84] satisfies linear targeted malleability (Definition A.6) in the generic group model. We begin by recalling the ElGamal encryption scheme where the message is encoded in the exponent.

Construction C.4 (ElGamal Encryption [ElG84])**.** Let $\mathsf{GroupGen}$ be a prime-order group generation algorithm. The ElGamal encryption scheme $\Pi_{\mathsf{Enc}} = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Add}, \mathsf{ImVer})$ with message space $\mathbb{F}_p$ is defined as follows:

- $\mathsf{Setup}(1^\lambda)$: On input the security parameter λ, the setup algorithm computes $(\mathbb{G}, p, g) \leftarrow \mathsf{GroupGen}(1^\lambda)$, samples $x \xleftarrow{\text{R}} \mathbb{F}_p$, and computes $h \leftarrow g^x$. It outputs the public key $\mathsf{pk} = (\mathbb{G}, p, g, h)$ and the secret key $\mathsf{sk} = (\mathbb{G}, p, g, x)$. The message space for the ElGamal encryption scheme is $\mathbb{F}_p$.
- $\mathsf{Encrypt}(\mathsf{pk}, a)$: On input the public key $\mathsf{pk} = (\mathbb{G}, p, g, h)$ and a message $a \in \mathbb{F}_p$, the encryption algorithm samples $r \xleftarrow{\text{R}} \mathbb{F}_p$ and outputs the ciphertext $\mathsf{ct} = (g^r, h^r g^a)$.
- $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}, \mathcal{M})$: On input a secret key $\mathsf{sk} = (\mathbb{G}, p, g, x)$, a ciphertext $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$ and a set of candidate messages $\mathcal{M} \subseteq \mathbb{F}_p$, the decryption algorithm checks whether there exists $a \in \mathcal{M}$ such that $g^a = \mathsf{ct}_2/\mathsf{ct}_1^x$. If so, it outputs a; otherwise, it outputs $\perp$.
- $\mathsf{Add}(\mathsf{pk}, \mathsf{ct}_1, \mathsf{ct}_2)$: On input the public key $\mathsf{pk} = (\mathbb{G}, p, g, h)$ and ciphertexts $\mathsf{ct}_1 = (g^{r_1}, h^{r_1} g^{a_1})$, $\mathsf{ct}_2 = (g^{r_2}, h^{r_2} g^{a_2})$, the addition algorithm outputs $\mathsf{ct}' = (g^{r_1} g^{r_2}, (h^{r_1} g^{a_1}) \cdot (h^{r_2} g^{a_2}))$.
- $\mathsf{ImVer}(\mathsf{sk}, \mathsf{ct}, \mathcal{M})$: On input a secret key sk, a ciphertext ct and a set of candidate messages $\mathcal{M} \subseteq \mathbb{F}_p$, the image-verification algorithm outputs 1 if $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}, \mathcal{M}) \neq \perp$ and 0 otherwise.

Theorem C.5 (Correctness and Security of ElGamal [ElG84])**.** *Construction C.4 is correct, and moreover, if the Decisional Diffie-Hellman (DDH) assumption holds with respect to* $\mathsf{GroupGen}$*, then Construction C.4 is semantically secure.*

Theorem C.6 (Linear Targeted Malleability of ElGamal)**.** *The ElGamal encryption scheme* Π_{Enc} *from Construction C.4 satisfies linear targeted malleability with respect to any target message space* $\mathcal{M}$ *where* $|\mathcal{M}| = \mathsf{poly}(\lambda)$ *(Definition A.6) in the generic group model.*

Proof. Take any efficient adversary $\mathcal{A}$ and plaintext generator $\mathcal{T}$ for the linear targeted malleability game. We construct an efficient simulator $\mathcal{S}$ that works as follows:

1. At the beginning of the security game, the simulator receives the public key pk and (possibly) auxiliary input z. In the generic group model, the public key consists of $\mathsf{pk} = (\mathsf{pp}, p, g, h)$ where $\mathsf{pp}, g, h \in \{0,1\}^\lambda$ are arbitrary bit strings. The simulator initializes an empty table T and adds mappings $g \mapsto 1$ and $h \mapsto \hat{x}$, where $\hat{x}$ is a formal variable (that represents the secret key of the encryption scheme, unknown to the simulator).

2. For each $i \in [m]$, the simulator samples $\mathsf{ct}_{i,1}, \mathsf{ct}_{i,2} \xleftarrow{\text{R}} \{0,1\}^\lambda$ and sets $\mathsf{ct}_i \leftarrow (\mathsf{ct}_{i,1}, \mathsf{ct}_{i,2})$. The simulator also adds mappings $\mathsf{ct}_{i,1} \mapsto \hat{r}_i$ and $\mathsf{ct}_{i,2} \mapsto \hat{r}_i\hat{x} + \hat{a}_i$, where $\hat{r}_1, \ldots, \hat{r}_m$ and $\hat{a}_1, \ldots, \hat{a}_m$ are formal variables representing the encryption randomness and the messages.

3. The simulator starts running the adversary $\mathcal{A}$ on input $\mathsf{pk} = (\mathsf{pp}, p, g, h)$, $\mathsf{ct}_1, \ldots, \mathsf{ct}_m$, and z. The simulator $\mathcal{S}$ will simulate the generic group oracle queries for $\mathcal{A}$ as follows:

   - $\mathsf{GGM.Setup}$: The simulator replies to $\mathcal{A}$ with $\perp$.
   - $\mathsf{GGM.Encode}$: The simulator replies to $\mathcal{A}$ with $\perp$.
   - $\mathsf{GGM.Add}$: On input a key k and handles $\xi_1, \xi_2 \in \{0,1\}^\lambda$, the simulator checks that $k = \mathsf{pp}$ and that ξ_1, ξ_2 are present in T and mapped to formal polynomials f_1, f_2 over $\mathbb{F}_p$ (returning $\perp$ otherwise). If the checks pass, the oracle samples a fresh handle $\xi \xleftarrow{\text{R}} \{0,1\}^\lambda$ and adds the entry $\xi \mapsto (f_1 + f_2)$ to T and replies with ξ.
   - $\mathsf{GGM.Test}$: On input a key k and a handle $\xi \in \{0,1\}^\lambda$, the simulator checks that $k = \mathsf{pp}$, that ξ is present in T, and mapped to a formal polynomial f over $\mathbb{F}_p$ (returning $\perp$ otherwise). If the checks pass, the oracle outputs "zero" if $f \equiv 0$ is the identically-zero polynomial over $\mathbb{F}_p$ and "nonzero" otherwise.

4. At the end of the experiment, the adversary outputs a collection of ciphertexts $\mathsf{ct}'_1, \ldots, \mathsf{ct}'_k$, where each $\mathsf{ct}'_i = (\mathsf{ct}'_{i,1}, \mathsf{ct}'_{i,2})$. The simulator first checks that each $\mathsf{ct}'_{i,1} \mapsto f_{i,1}$ and $\mathsf{ct}'_{i,2} \mapsto f_{i,2}$ in T. If this is not the case for any $i \in [k]$, the simulator outputs $\perp$.

5. By construction of the simulator, every $f_{i,1}$ and $f_{i,2}$ can be expressed as a linear function in the variables $\hat{r}_1, \ldots, \hat{r}_m, \hat{a}_1, \ldots, \hat{a}_m, \hat{x}$. The simulator now checks that $f_{i,2} - \hat{x} f_{i,1} \equiv \left(\sum_{j \in [m]} \alpha_{i,j} \hat{a}_j\right) + \beta_i$, for some choice of scalars $\alpha_{i,1}, \ldots, \alpha_{i,m}, \beta_i \in \mathbb{F}_p$. If this relation does not hold for any $i \in [k]$, then the simulator outputs $\perp$. Otherwise, the simulator sets the i^{th} row of $\mathbf{\Pi} \in \mathbb{F}_p^{k \times m}$ to $[\alpha_{i,1}, \ldots, \alpha_{i,m}]$ and the i^{th} component of $\mathbf{b} \in \mathbb{F}_p^k$ to β_i.

6. Output the matrix $\mathbf{\Pi} \in \mathbb{F}_p^{k \times m}$ and the vector $\mathbf{b} \in \mathbb{F}_p^k$.

We now show that the real distribution and the simulated distributions are computationally indistinguishable when $\mathsf{GroupGen}$ is modeled as a generic group $\mathcal{G}$. To do so, we use a hybrid argument:

- Hyb_0: This is the real distribution in Definition A.6. Namely, the experiment proceeds as follows:

  1. The challenger samples $(\mathsf{pp}, \mathsf{sk}', p) \leftarrow \mathsf{GGM.Setup}(1^\lambda)$, $g \leftarrow \mathsf{GGM.Encode}(\mathsf{sk}, 1)$, $x \xleftarrow{\text{R}} \mathbb{F}_p$, and $h \leftarrow \mathsf{GGM.Encode}(\mathsf{sk}', x)$. It then sets $\mathsf{pk} = (\mathsf{pp}, p, g, h)$ and $\mathsf{sk} = (\mathsf{pp}, p, g, \mathsf{sk}', x)$.
  2. The challenger computes $(s, a_1, \ldots, a_m) \leftarrow \mathcal{T}(\mathsf{pk})$ and $\mathsf{ct}_i \leftarrow \mathsf{Encrypt}(\mathsf{pk}, a_i)$ for each $i \in [m]$. Namely, for each $i \in [m]$, the challenger samples $r_i \xleftarrow{\text{R}} \mathbb{F}_p$, and computes $\mathsf{ct}_{i,1}$ to be an encoding of r_i and $\mathsf{ct}_{i,2}$ to be an encoding of $r_i x + a_i$ (using the public components pp, g, h and queries to $\mathsf{GGM.Add}$). It sets $\mathsf{ct}_i = (\mathsf{ct}_{i,1}, \mathsf{ct}_{i,2})$. In particular, $\mathsf{ct}_{i,1} \mapsto r_i$ and $\mathsf{ct}_{i,2} \mapsto r_i x + a_i$ in the internal table T maintained by the generic group oracle.

3. The challenger sends the public key pk, the ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_m$, and the auxiliary information z to the adversary $\mathcal{A}$. The adversary is also given access to the generic group oracles $\mathsf{GGM.Setup}$, $\mathsf{GGM.Encode}$, $\mathsf{GGM.Add}$, $\mathsf{GGM.Test}$.
4. At some point, the adversary outputs $\mathsf{ct}'_1, \ldots, \mathsf{ct}'_k$.
5. If $\mathsf{ImVer}(\mathsf{sk}, \mathsf{ct}'_i, \mathcal{M}) \neq 1$ for any $i \in [k]$, the challenger outputs $\perp$. In the case of the ElGamal scheme, this means the challenger outputs $\perp$ unless $\mathsf{ct}'_i = (\mathsf{ct}'_{i,1}, \mathsf{ct}'_{i,2})$ where $\mathsf{ct}'_{i,1} \mapsto r'_i$ and $\mathsf{ct}'_{i,2} \mapsto r'_i x + a'_i$ in the internal table T maintained by the generic group oracle for some $r'_i \in \mathbb{F}_p$ and $a'_i \in \mathcal{X}$. If ct'_i satisfy this requirement for all $i \in [k]$, then the output of the experiment is $(\mathsf{pk}, \{a_i\}_{i\in[m]}, s, \{a'_i\}_{i\in[k]})$.

- Hyb_1: Same as Hyb_0 except the challenger computes the ciphertext components $\mathsf{ct}_{i,1}, \mathsf{ct}_{i,2}$ by directly encoding the value of r_i and $r_i x + a_i$, respectively, using the secret key sk'. In other words, the challenger computes $\mathsf{ct}_{i,1} \leftarrow \mathsf{GGM.Encode}(\mathsf{sk}', r_i)$ and $\mathsf{ct}_{i,2} \leftarrow \mathsf{GGM.Encode}(\mathsf{sk}', r_i x + a_i)$.

- Hyb_2: Same as Hyb_1, except the challenger implements the generic group oracles according to the specification of $\mathcal{S}$. In particular, the challenger proceeds as follows in this experiment:

  1. The challenger runs $(\mathsf{pp}, \mathsf{sk}', p) \leftarrow \mathsf{GGM.Setup}(1^\lambda)$. Then, it samples $x \xleftarrow{\mathrm{R}} \mathbb{F}_p$, initializes an empty table T, samples two strings $g, h \xleftarrow{\mathrm{R}} \{0,1\}^\lambda$, and adds the mappings $g \mapsto 1$ and $h \mapsto \hat{x}$ to T. Here $\hat{x}$ is a *formal variable* representing the secret key. The challenger sets $\mathsf{pk} = (\mathsf{pp}, p, g, h)$.
  2. The challenger computes $(s, a_1, \ldots, a_m) \leftarrow \mathcal{T}(\mathsf{pk})$. For each $i \in [m]$, it samples $r_i \xleftarrow{\mathrm{R}} \mathbb{F}_p$ and $\mathsf{ct}_{i,1}, \mathsf{ct}_{i,2} \xleftarrow{\mathrm{R}} \{0,1\}^\lambda$. It adds mappings $\mathsf{ct}_{i,1} \mapsto \hat{r}_i$ and $\mathsf{ct}_{i,2} \mapsto \hat{r}_i \hat{x} + \hat{a}_i$, where $\hat{r}_1, \ldots, \hat{r}_m$ and $\hat{a}_1, \ldots, \hat{a}_m$ are formal variables for the encryption randomness and the message, respectively.
  3. The challenger sends the public key pk, the ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_m$, and the auxiliary information z to $\mathcal{A}$. It then responds to the generic group oracle queries using the same procedure as the simulator $\mathcal{S}$ (with respect to its own table T).
  4. At the end of the experiment, the adversary outputs ciphertexts $\mathsf{ct}'_1, \ldots, \mathsf{ct}'_k$.
  5. The challenger now scans through the values in T and instantiates each of the formal variables $\hat{x}, \hat{r}_1, \ldots, \hat{r}_m, \hat{a}_1, \ldots, \hat{a}_m$ with their real values $x, r_1, \ldots, r_m, a_1, \ldots, a_m$, respectively. The output of Hyb_2 is then computed using the same procedure as in Hyb_1.

- Hyb_3: Same as Hyb_2 except after the adversary outputs $\mathsf{ct}'_1, \ldots, \mathsf{ct}'_k$, instantiating the formal variables $\hat{x}, \hat{r}_1, \ldots, \hat{r}_m, \hat{a}_1, \ldots, \hat{a}_m$ with their values $x, r_1, \ldots, r_m, a_1, \ldots, a_m$, and verifying $\mathsf{ImVer}(\mathsf{sk}, \mathsf{ct}'_i, \mathcal{M}) = 1$ for all $i \in [k]$, the challenger computes the output using the following procedure:

  1. The challenger parses ct'_i as $(\mathsf{ct}'_{i,1}, \mathsf{ct}'_{i,2})$, and checks that $\mathsf{ct}'_{i,1} \mapsto f_{i,1}$ and $\mathsf{ct}_{i,2} \mapsto f_{i,2}$ in T. It then checks that $f_{i,2} - \hat{x} f_{i,1} \equiv \left(\sum_{j\in[m]} \alpha_{i,j} \hat{a}_j\right) + \beta_i$, for some choice of scalars $\alpha_{i,1}, \ldots, \alpha_{i,m}, \beta_i \in \mathbb{F}_p$. If this relation does not hold for any $i \in [k]$, then the challenger outputs $\perp$.
  2. Let $\mathbf{\Pi} \in \mathbb{F}_p^{k\times m}$ be the matrix whose i^{th} row is $[\alpha_{i,1}, \ldots, \alpha_{i,m}]$ and let $\mathbf{b} \in \mathbb{F}_p^k$ be the vector whose i^{th} component is β_i. The challenger computes $(a'_1, \ldots, a'_k)^\mathsf{T} \leftarrow \mathbf{\Pi} \cdot (a_1, \ldots, a_m) + \mathbf{b}$.
  3. Output the tuple $(\mathsf{pk}, \{a_i\}_{i\in[m]}, s, \{a'_i\}_{i\in[k]})$.

  This is the ideal distribution with simulator $\mathcal{S}$ in Definition A.6.

For an adversary $\mathcal{A}$, we write $\mathsf{Hyb}_i(\mathcal{A})$ to denote the output of an execution of the game described by Hyb_i with adversary $\mathcal{A}$. We now show that each adjacent pair of hybrids are statistically indistinguishable to any adversary that makes polynomially-many queries to the generic group oracle.

Lemma C.7. *For all adversaries $\mathcal{A}$ making $\mathsf{poly}(\lambda)$ queries to $\mathcal{G}$, $\mathsf{Hyb}_0(\mathcal{A}) \overset{s}{\approx} \mathsf{Hyb}_1(\mathcal{A})$.*

Proof. The only difference between Hyb_0 and Hyb_1 is that the challenger uses $\mathsf{GGM.Add}$ to construct the ciphertext components $\mathsf{ct}_{i,1}$ and $\mathsf{ct}_{i,2}$ rather than $\mathsf{GGM.Encode}$. Let $t = \mathsf{poly}(\lambda)$ be the number of queries to $\mathsf{GGM.Add}$ the challenger makes to $\mathsf{GGM.Add}$ to compute $\mathsf{ct}_1, \ldots, \mathsf{ct}_m$. The only difference between Hyb_0 and Hyb_1 is that computing $\mathsf{ct}_1, \ldots, \mathsf{ct}_m$ introduces t entries into the generic group table T in Hyb_0 while computing $\mathsf{ct}_1, \ldots, \mathsf{ct}_m$ introduces $2m$ entries into the T in Hyb_1. However, these additional entries in T are information-theoretically hidden from the view of $\mathcal{A}$, and the view of $\mathcal{A}$ is *identically* distributed in Hyb_0 and Hyb_1 unless $\mathcal{A}$ makes a query to $\mathcal{G}$ on an encoding present in T in Hyb_0 but not in Hyb_1. There are at most t such encodings. Since the encodings are uniform over $\{0,1\}^\lambda$, and the adversary makes $Q = \mathsf{poly}(\lambda)$ queries, the probability that the adversary makes a query on one of these encodings is at most $Qt/2^\lambda = \mathsf{negl}(\lambda)$. Thus, the view of $\mathcal{A}$ in Hyb_0 and Hyb_1 is statistically indistinguishable and the claim holds. □

Lemma C.8. *For all adversaries $\mathcal{A}$ making $\mathsf{poly}(\lambda)$ queries to $\mathcal{G}$, $\mathsf{Hyb}_1(\mathcal{A}) \stackrel{s}{\approx} \mathsf{Hyb}_2(\mathcal{A})$.*

Proof. We show that the view of the adversary $\mathcal{A}$ is statistically indistinguishable in the two experiments. By construction, the public key $\mathsf{pk} = (\mathsf{pp}, p, g, h)$, the ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_m$, and the auxiliary data z are identically distributed in Hyb_1 and Hyb_2. Namely, in both experiments, pp, g, h are uniform random strings over $\{0,1\}^\lambda$, and each ciphertext ct_i consists of two uniformly random strings over $\{0,1\}^\lambda$. It suffices to argue that each of the adversary's queries to the generic group oracle are statistically indistinguishable. We use a hybrid argument over the number of queries the adversary makes, where in the i^{th} hybrid $\mathsf{Hyb}_{1,i}$, the first i queries are answered according to the specification in Hyb_2 while the remaining queries are answered according to the specification in Hyb_1. We show that for all i, the outputs of $\mathsf{Hyb}_{1,i-1}$ and $\mathsf{Hyb}_{1,i}$ are statistically indistinguishable. It suffices to consider the i^{th} query:

- $\mathsf{GGM.Setup}$: In both $\mathsf{Hyb}_{1,i-1}$ and $\mathsf{Hyb}_{1,i}$, the setup oracle outputs $\perp$.
- $\mathsf{GGM.Encode}$: In $\mathsf{Hyb}_{1,i-1}$, the $\mathsf{GGM.Encode}$ oracle outputs $\perp$ unless the adversary queries the oracle on $k = \mathsf{sk}'$. Since the view of $\mathcal{A}$ on the first $i-1$ queries in $\mathsf{Hyb}_{1,i-1}$ is independent of sk' and sk' is uniform over $\{0,1\}^\lambda$, the output in $\mathsf{Hyb}_{1,i-1}$ is $\perp$ with probability $1 - 2^{-\lambda}$. In $\mathsf{Hyb}_{1,i}$, the output is $\perp$ with probability 1, so the output distribution on the i^{th} query is statistically indistinguishable in these two experiments.
- $\mathsf{GGM.Add}$: The addition oracle has identical behavior in $\mathsf{Hyb}_{1,i-1}$ and $\mathsf{Hyb}_{1,i}$. Namely, if the adversary provides $k = \mathsf{pp}$ and two valid handles ξ_1, ξ_2, then it receives a uniformly random string in $\{0,1\}^\lambda$ as its output, and if it provides an invalid input, it receives $\perp$.
- $\mathsf{GGM.Test}$: Suppose an adversary makes a query to $\mathsf{GGM.Test}$ on $k = \mathsf{pp}$ and $\xi \in \mathsf{T}$. On all other queries, the oracle outputs $\perp$ in both experiments. This means that ξ must have been added to T as a result of a $\mathsf{GGM.Encode}$ query or as a result of a $\mathsf{GGM.Add}$ query. By construction of $\mathsf{Hyb}_{1,i+1}$, this means that
$$\xi \mapsto \Bigg[\underbrace{\alpha + \beta\hat{x} + \sum_{i\in[m]} (\gamma_i \hat{r}_i + \delta_i(\hat{r}_i\hat{x} + \hat{a}_i))}_{f(\hat{x},\hat{r}_1,\ldots,\hat{r}_m,\hat{a}_1,\ldots,\hat{a}_m)}\Bigg], \tag{C.1}$$
for some choice of scalars $\alpha, \beta, \gamma_1, \ldots, \gamma_m, \delta_1, \ldots, \delta_m \in \mathbb{F}_p$. By construction, the output of $\mathsf{GGM.Test}$ in $\mathsf{Hyb}_{1,i-1}$ is 1 if and only if $f(x, r_1, \ldots, r_m, a_1, \ldots, a_m) = 0$ where $x, r_1, \ldots, r_m, a_1, \ldots, a_m \in \mathbb{F}_p$ are the scalars sampled by the challenger in the experiment. In $\mathsf{Hyb}_{1,i}$, the output of $\mathsf{GGM.Test}$ is 1 only if $f \equiv 0$ is the identically-zero polynomial (in the formal variables $\hat{x}, \hat{r}_1, \ldots, \hat{r}_m, \hat{a}_1, \ldots, \hat{a}_m$). Thus, these two experiments only differ if $f \not\equiv 0$, but $f(x, r_1, \ldots, r_m, a_1, \ldots, a_m) \neq 0$. We show that this happens with negligible probability. First, in both $\mathsf{Hyb}_{1,i-1}$ and $\mathsf{Hyb}_{1,i}$, the first $i-1$ queries are handled according to the specification in Hyb_2. By construction, this means that everything in the experiment prior to the i^{th} query can be simulated *without* knowledge of the value of $x, r_1, \ldots, r_m$, and these values can in fact be sampled *after* the adversary has submitted its i^{th} query (i.e., after the adversary has

committed to the polynomial f). First, define the polynomial g in the formal variables $\hat{x}, \hat{r}_1, \ldots, \hat{r}_m$ to be the polynomial f where each of the variables $\hat{a}_i$ are instantiated with their actual value $a_i \in \mathbb{F}_p$:

$$\begin{aligned} g(\hat{x}, \hat{r}_1, \ldots, \hat{r}_m) &= f(\hat{x}, \hat{r}_1, \ldots, \hat{r}_m, a_1, \ldots, a_m) \\ &= \left(\alpha + \sum_{i \in [m]} \delta_i a_i\right) + \beta \hat{x} + \sum_{i \in [m]} (\gamma_i \hat{r}_i + \delta_i \hat{r}_i \hat{x}) \end{aligned}$$

It is easy to see that if $f \not\equiv 0$, then $g \not\equiv 0$ (since if $g \equiv 0$, then it must be the case that $\beta, \gamma_i, \delta_i = 0$ for all $i \in [m]$, which correspondingly means that $\alpha = 0$, in which case $f \equiv 0$). Since g is a polynomial of total degree 2 in the variables $\hat{x}, \hat{r}_1, \ldots, \hat{r}_m$, we conclude by the Schwartz-Zippel lemma that

$$\Pr[g(x, r_1, \ldots, r_m) = 0 \mid x, r_1, \ldots, r_m \xleftarrow{R} \mathbb{F}_p] \leq 2/p = \mathsf{negl}(\lambda).$$

Thus, with overwhelming probability over the choice of $x, r_1, \ldots, r_m$, if $f \not\equiv 0$, then

$$f(x, r_1, \ldots, r_m, a_1, \ldots, a_m) = g(x, r_1, \ldots, r_m) \neq 0,$$

in which case the output in $\mathsf{Hyb}_{1,i-1}$ is 0, which coincides with the output in $\mathsf{Hyb}_{1,i}$.

Since the adversary makes a polynomial number of queries to the generic group oracle, and the response to each query is statistically indistinguishable, we conclude via a hybrid argument that the adversary's output in the two distributions are statistically indistinguishable. Since the output of the experiment is computed using the same procedure in Hyb_1 and Hyb_2, we conclude that the outputs of the two distributions are also statistically indistinguishable. □

Lemma C.9. *For all adversaries $\mathcal{A}$, $\mathsf{Hyb}_2(\mathcal{A}) \stackrel{s}{\approx} \mathsf{Hyb}_3(\mathcal{A})$.*

Proof. By construction, the view of $\mathcal{A}$ in Hyb_2 and Hyb_3 is identically distributed. Thus, the ciphertexts $\mathsf{ct}'_i, \ldots, \mathsf{ct}'_k$ output by $\mathcal{A}$ in the two experiments are identically distributed. It suffices to argue that the output in the two experiments are statistically indistinguishable. We write $\mathsf{ct}'_i = (\mathsf{ct}'_{i,1}, \mathsf{ct}'_{i,2})$ where each $\mathsf{ct}'_{i,1}, \mathsf{ct}'_{i,2} \in \{0,1\}^\lambda$. In addition, we note that in both Hyb_2 and Hyb_3, all of the interactions with $\mathcal{A}$ can be implemented without knowledge of the values of $x, r_1, \ldots, r_m$. Thus, we can defer the sampling of $x, r_1, \ldots, r_m \xleftarrow{R} \mathbb{F}_p$ until *after* the adversary has output $\mathsf{ct}'_1, \ldots, \mathsf{ct}'_k$. We consider several possibilities:

- Suppose that there exists $i \in [k]$ where $\mathsf{ImVer}(\mathsf{sk}, \mathsf{ct}'_i, \mathcal{M}) = 0$. Then, both experiments output $\perp$.
- Suppose that there exists $i \in [k]$ where $\mathsf{ct}'_{i,1} \notin \mathsf{T}$ or $\mathsf{ct}'_{i,2} \notin \mathsf{T}$. Then, $\mathsf{ImVer}(\mathsf{sk}, \mathsf{ct}'_i, \mathcal{M}) = 0$, and both experiments output $\perp$.
- Suppose $\mathsf{ct}'_{i,1} \mapsto f_{i,1}$ and $\mathsf{ct}'_{i,2} \mapsto f_{i,2}$ in T, but $f_{i,2} - \hat{x} f_{i,1} \not\equiv \left(\sum_{j \in [m]} \alpha_{i,j} \hat{a}_j\right) + \beta_i$ for some $i \in [k]$. We argue that with overwhelming probability, $\mathsf{ImVer}(\mathsf{sk}, \mathsf{ct}'_i, \mathcal{M}) = 0$ and both experiments output $\perp$. Take any $a \in \mathcal{M} \subseteq \mathbb{F}_p$ and define the polynomial g_a in the formal variables $\hat{x}, \hat{r}_1, \ldots, \hat{r}_m$:

  $$g_a(\hat{x}, \hat{r}_1, \ldots, \hat{r}_m) = f_{i,2}(\hat{x}, \hat{r}_1, \ldots, \hat{r}_m, a_1, \ldots, a_m) - \hat{x} \cdot f_{i,1}(\hat{x}, \hat{r}_1, \ldots, \hat{r}_m, a_1, \ldots, a_m) - a.$$

  By the structure of $f_{i,1}$ and $f_{i,2}$ (Eq. (C.1)) and the assumption that $f_{i,2} - \hat{x} f_{i,1} \not\equiv \left(\sum_{j \in [m]} \alpha_{i,j} \hat{a}_j\right) + \beta_i$, we have that $g_a \not\equiv 0$. Since g_a is a polynomial of total degree 3, we can appeal to the the Schwartz-Zippel lemma and conclude that for any $a \in \mathcal{M}$,

  $$\Pr[g_a(x, r_1, \ldots, r_m) = 0 \mid x, r_1, \ldots, r_m \xleftarrow{R} \mathbb{F}_p] = 3/p.$$

  By a union bound,

  $$\Pr[\exists a \in \mathcal{M} : g_a(x, r_1, \ldots, r_m) = 0 \mid x, r_1, \ldots, r_m \xleftarrow{R} \mathbb{F}_p] = 3|\mathcal{M}|/p = \mathsf{negl}(\lambda),$$

since $|\mathcal{M}| = \mathsf{poly}(\lambda)$. Thus, with overwhelming probability over the choice of $x, r_1, \ldots, r_m$, we have that $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}'_i, \mathcal{M}) = \bot$. Correspondingly, with overwhelming probability, $\mathsf{ImVer}(\mathsf{sk}, \mathsf{ct}'_i, \mathcal{M}) = 0$ and both experiments output $\bot$ in this case.

- Suppose $\mathsf{ct}'_{i,1} \mapsto f_{i,1}$ and $\mathsf{ct}'_{i,2} \mapsto f_{i,2}$ in T where $f_{i,2} - \hat{x} f_{i,1} \equiv \left(\sum_{j \in [m]} \alpha_{i,j} \hat{a}_j\right) + \beta_i$, and moreover, that $\mathsf{ImVer}(\mathsf{sk}, \mathsf{ct}'_i, \mathcal{M}) = 1$ for all $i \in [k]$. Since $\mathsf{ImVer}(\mathsf{sk}, \mathsf{ct}', \mathcal{M}) = 1$, there exists $a'_i \in \mathcal{M}$ such that

$$f_{i,2}(x, r_1, \ldots, r_m, a_1, \ldots, a_m) - x f_1(x, r_1, \ldots, r_m, a_1, \ldots, a_m) = a'_i.$$

  Since $f_{i,2} - \hat{x} f_{i,1} \equiv \left(\sum_{j \in [m]} \alpha_{i,j} \hat{a}_j\right) + \beta_i$, this means that

$$f_{i,2}(x, r_1, \ldots, r_m, a_1, \ldots, a_m) - x f_1(x, r_1, \ldots, r_m, a_1, \ldots, a_m) = \sum_{j \in [m]} \alpha_{i,j} a_j + \beta_i = a'_i.$$

  Since this holds for all $i \in [k]$, this means that $(a'_1, \ldots, a'_k)^\mathsf{T} = \mathbf{\Pi} \cdot (a_1, \ldots, a_m)^\mathsf{T} + \mathbf{b}$, where $\mathbf{\Pi} \in \mathbb{F}_p^{k \times m}$ and $\mathbf{b} \in \mathbb{F}_p^k$ are precisely the quantities the challenger constructs in Hyb_3. Thus, in this case, the output in Hyb_2 and Hyb_3 are identically distributed. □

The claim now follows by combining Lemmas C.7 to C.9. □

C.2 Adaptive Soundness of Construction A.8 in the Generic Group Model

In this section, we show that Construction A.8 satisfies adaptive soundness when the underlying encryption scheme is instantiated with ElGamal encryption over a generic group (Construction C.4).

Theorem C.10. *Let $\mathcal{C} = \{C_\ell\}_{\ell \in \mathbb{N}}$ be a family of arithmetic circuits. Let Π_{Enc} be the ElGamal encryption (Construction C.4) scheme with plaintext space $\mathbb{F}_p$, and let Π_{LPCP} be a 1-query linear PCP over $\mathbb{F}_p$ for the relation $\mathcal{R}_\mathcal{C}$ with bound function $B = B(\lambda)$, query length $\ell = \ell(\lambda)$ and soundness error $\varepsilon = \varepsilon(\lambda)$. Then Construction A.8 is a preprocessing SNARG for $\mathcal{R}_\mathcal{C}$ with perfect completeness and adaptive soundness error $\varepsilon + \mathsf{negl}(\lambda)$ when the group $\mathbb{G}$ in the ElGamal encryption scheme is modeled as a generic group.*

Proof. Completeness follows exactly as in Theorem A.9, so it suffices to argue adaptive soundness. We use a hybrid argument similar to that used in the proof of Theorem C.6. As in the proof of Theorem C.6, we model $\mathsf{GroupGen}$ as a generic group $\mathcal{G}$.

- Hyb_0: This is the adaptive soundness experiment:
  1. The challenger begins by sampling $(\mathsf{pp}, \mathsf{sk}, p) \leftarrow \mathsf{GGM.Setup}(1^\lambda)$, $g \leftarrow \mathsf{GGM.Encode}(\mathsf{sk}, 1)$, $z \xleftarrow{\mathrm{R}} \mathbb{F}_p$, and $h \xleftarrow{\mathrm{R}} \mathsf{GGM.Encode}(\mathsf{sk}, z)$. It sets $\mathsf{pk} = (\mathsf{pp}, p, g, h)$. Next, the challenger samples $(\mathsf{st}, \mathbf{q}_{\mathsf{inp}}, \mathbf{q}) \leftarrow \mathcal{Q}_{\mathsf{LPCP}}$ where $\mathbf{q}_{\mathsf{inp}} \in \mathbb{F}_p^n$ and $\mathbf{q} = (q_1, \ldots, q_\ell) \in \mathbb{F}_p^\ell$. Then, the setup algorithm computes the ciphertexts $\mathsf{ct}_i \leftarrow \mathsf{Encrypt}(\mathsf{pk}, q_i)$ for each $i \in [\ell]$. For each $i \in [\ell]$, the challenger samples $r_i \xleftarrow{\mathrm{R}} \mathbb{F}_p$ and computes $\mathsf{ct}_{i,1}$ to be an encoding of r_i and $\mathsf{ct}_{i,2}$ to be an encoding of $r_i z + q_i$ (using the public components pp, g, h and queries to $\mathsf{GGM.Add}$). It sets $\mathsf{ct}_i = (\mathsf{ct}_{i,1}, \mathsf{ct}_{i,2})$. In particular, $\mathsf{ct}_{i,1} \mapsto r_i$ and $\mathsf{ct}_{i,2} \mapsto r_i z + q_i$ in the internal table T maintained by the generic group oracle.
  2. The challenger gives $\mathsf{crs} = (\mathsf{pk}, \{\mathsf{ct}_i\}_{i \in [\ell]})$ to the adversary $\mathcal{A}$. In addition, the adversary is given access to the generic group oracles $\mathsf{GGM.Setup}, \mathsf{GGM.Encode}, \mathsf{GGM.Add}, \mathsf{GGM.Test}$.
  3. At some point, the adversary outputs a statement $\mathbf{x}$ and a proof π. If $\mathbf{x} \in \mathcal{L}_\mathcal{C}$, then the challenger outputs 0. Otherwise, the challenger parses $\pi = (\mathsf{ct}'_1, \mathsf{ct}'_2)$ and checks that $\mathsf{ct}'_1 \mapsto r$ and $\mathsf{ct}'_2 \mapsto rz + a$ for some $r \in \mathbb{F}_p$ and $a \in [-B, B]$ in the internal table T maintained by the generic group oracle. If not, the challenger outputs 0. Otherwise, the challenger outputs $\mathcal{V}_{\mathsf{LPCP}}(\mathsf{st}, \mathbf{q}_{\mathsf{inp}}^\mathsf{T} \mathbf{x}, a)$.
- Hyb_1: Same as Hyb_0 except the challenger computes the ciphertext components $\mathsf{ct}_{i,1}, \mathsf{ct}_{i,2}$ by directly encoding the value of r_i and $r_i z + q_i$ using the secret key sk. Specifically, the challenger computes $\mathsf{ct}_{i,1} \leftarrow \mathsf{GGM.Encode}(\mathsf{sk}, r_i)$ and $\mathsf{ct}_{i,2} \leftarrow \mathsf{GGM.Encode}(\mathsf{sk}, r_i z + q_i)$.

- Hyb_2: Same as Hyb_1 except the challenger implements the generic group oracle queries with the following modified procedure:

  1. The challenger starts by running $(\mathsf{pp}, \mathsf{sk}, p) \leftarrow \mathsf{GGM.Setup}(1^\lambda)$ and samples $z \xleftarrow{R} \mathbb{F}_p$. It then initializes an empty table T, samples two strings $g, h \xleftarrow{R} \{0,1\}^\lambda$ and adds the mappings $g \mapsto 1$ and $h \mapsto \hat{z}$ to T. Here, $\hat{z}$ is a *formal variable* representing the secret key. The challenger sets $\mathsf{pk} = (\mathsf{pp}, p, g, h)$.
  2. Next, for each $i \in [\ell]$, the challenger samples $r_i \xleftarrow{R} \mathbb{F}_p$, and $\mathsf{ct}_{i,1}, \mathsf{ct}_{i,2} \xleftarrow{R} \{0,1\}^\lambda$. It adds mappings $\mathsf{ct}_{i,1} \mapsto \hat{r}_i$ and $\mathsf{ct}_{i,2} \mapsto \hat{r}_i\hat{z} + \hat{q}_i$ where $\hat{r}_i, \ldots, \hat{r}_\ell$ and $\hat{q}_1, \ldots, \hat{q}_\ell$ are formal variables for the encryption randomness and the query component, respectively. It still sets $\mathsf{ct}_i = (\mathsf{ct}_{i,1}, \mathsf{ct}_{i,2})$ as before.
  3. The challenger gives $\mathsf{crs} = (\mathsf{pk}, \{\mathsf{ct}_i\}_{i \in [\ell]})$ to the adversary. It then responds to the adversary's generic group queries as follows:
     - $\mathsf{GGM.Setup}$: The challenger replies to $\mathcal{A}$ with $\perp$.
     - $\mathsf{GGM.Encode}$: The challenger replies to $\mathcal{A}$ with $\perp$.
     - $\mathsf{GGM.Add}$: On input a key k and handles $\xi_1, \xi_2 \in \{0,1\}^\lambda$, the simulator checks that $k = \mathsf{pp}$ and that ξ_1, ξ_2 are present in T and mapped to formal polynomials f_1, f_2 over $\mathbb{F}_p$ (returning $\perp$ otherwise). If the checks pass, the oracle samples a fresh handle $\xi \xleftarrow{R} \{0,1\}^\lambda$ and adds the entry $\xi \mapsto (f_1 + f_2)$ to T and replies with ξ.
     - $\mathsf{GGM.Test}$: On input a key k and a handle $\xi \in \{0,1\}^\lambda$, the simulator checks that $k = \mathsf{pp}$, that ξ is present in T, and mapped to a formal polynomial f over $\mathbb{F}_p$ (returning $\perp$ otherwise). If the checks pass, the oracle outputs "zero" if $f \equiv 0$ is the identically-zero polynomial over $\mathbb{F}_p$ and "nonzero" otherwise.
  4. At the end of the experiment, the adversary outputs a statement $\mathbf{x}$ and a proof π. If $\mathbf{x} \notin \mathcal{L}_\mathcal{C}$, then the adversary outputs 0. Otherwise, the challenger scans through the values in T and instantiates each of the formal variables $\hat{z}, \hat{r}_1, \ldots, \hat{r}_\ell, \hat{q}_1, \ldots, \hat{q}_\ell$ with their real values $z, r_1, \ldots, r_\ell, q_1, \ldots, q_\ell$, respectively. The output of Hyb_2 is computed using the same procedure as in Hyb_1.

For an adversary $\mathcal{A}$, we write $\mathsf{Hyb}_i(\mathcal{A})$ to denote the output of experiment Hyb_i with adversary $\mathcal{A}$. We now show that the output distribution of any two adjacent hybrid experiments is statistically indistinguishable. Finally, we show that the output of $\mathsf{Hyb}_2(\mathcal{A})$ is 0 with probability negligibly close to ε.

Lemma C.11. *For all adversaries $\mathcal{A}$ making $\mathsf{poly}(\lambda)$ queries to $\mathcal{G}$, $\mathsf{Hyb}_0(\mathcal{A}) \overset{s}{\approx} \mathsf{Hyb}_1(\mathcal{A})$.*

Proof. Follows by a same analysis as the proof of Lemma C.7. □

Lemma C.12. *For all adversaries $\mathcal{A}$ making $\mathsf{poly}(\lambda)$ queries to $\mathcal{G}$, $\mathsf{Hyb}_1(\mathcal{A}) \overset{s}{\approx} \mathsf{Hyb}_2(\mathcal{A})$.*

Proof. Follows by a similar analysis as the proof of Lemma C.8. □

Lemma C.13. *If Π_{LPCP} is sound, then $\Pr[\mathsf{Hyb}_2(\mathcal{A}) = 1] \leq \varepsilon(\lambda) + \mathsf{negl}(\lambda)$.*

Proof. This follows by a similar analysis as the proof of Lemma C.9. First, we note that in Hyb_2, the adversary's view is entirely independent of the linear PCP query $\mathbf{q} \in \mathbb{F}_p^\ell$. This means that the challenger can sample $(\mathsf{st}, \mathbf{q}_{\mathsf{inp}}, \mathbf{q})$ *after* the adversary outputs the statement $\mathbf{x}$ and proof π. Similarly, the adversary's view is also independent of the value of the secret key z as well as the encryption randomness $r_1, \ldots, r_\ell$, so the challenger can also sample these values after the adversary has chosen its output. Write $\pi = (\mathsf{ct}'_1, \mathsf{ct}'_2)$. We consider several possibilities:

- If the adversary outputs a statement $\mathbf{x} \in \mathcal{L}$, then the output of the experiment is 0.
- If either $\mathsf{ct}'_1 \notin \mathsf{T}$ or $\mathsf{ct}'_2 \notin \mathsf{T}$, then the experiment outputs 0.

- Suppose $\mathsf{ct}'_1 \mapsto f_1$ and $\mathsf{ct}'_2 \mapsto f_2$ in T, but $f_2 - \hat{z} f_1 \not\equiv \left(\sum_{j\in[\ell]} \alpha_j \hat{q}_j\right) + \beta$, for some set of scalars $\alpha_1, \ldots, \alpha_\ell, \beta \in \mathbb{F}_p$. By construction of Hyb_2, every encoding ξ in T is a formal polynomial of the following form:

$$\xi \mapsto \Bigg[\underbrace{\bar{\alpha} + \bar{\beta}\hat{z} + \sum_{j\in[\ell]} (\bar{\gamma}_j \hat{r}_j + \bar{\delta}_j(\hat{r}_j \hat{x} + \hat{q}_j))}_{f(\hat{z}, \hat{r}_1, \ldots, \hat{r}_\ell, \hat{q}_1, \ldots, \hat{q}_\ell)}\Bigg],$$

  for some choice of scalars $\bar{\alpha}, \bar{\beta}, \bar{\gamma}_1, \ldots, \bar{\gamma}_\ell, \bar{\delta}_1, \ldots, \bar{\delta}_\ell \in \mathbb{F}_p$. We argue that with overwhelming probability over the choice of $z, r_1, \ldots, r_\ell$, the output of the experiment is 0 in this case. Take any $a \in [-B, B]$, and define the polynomial g_a in the formal variables $\hat{z}, \hat{r}_1, \ldots, \hat{r}_m$:

$$g_a(\hat{z}, \hat{r}_1, \ldots, \hat{r}_m) = f_2(\hat{z}, \hat{r}_1, \ldots, \hat{r}_\ell, q_1, \ldots, q_m) - \hat{z} \cdot f_1(\hat{z}, \hat{r}_1, \ldots, \hat{r}_\ell, q_1, \ldots, q_m) - a.$$

  Since $f_2 - \hat{z} f_1 \not\equiv \left(\sum_{j\in[\ell]} \alpha_j \hat{q}_j\right) + \beta$, we have that $g_a \not\equiv 0$. Since g_a is a polynomial of total degree 3, we can appeal to the the Schwartz-Zippel lemma and conclude that for any $a \in [-B, B]$,

$$\Pr[g_a(z, r_1, \ldots, r_\ell) = 0 \mid z, r_1, \ldots, r_\ell \xleftarrow{R} \mathbb{F}_p] = 3/p.$$

  By a union bound,

$$\Pr[\exists a \in [-B, B] : g_a(z, r_1, \ldots, r_\ell) = 0 \mid z, r_1, \ldots, r_\ell \xleftarrow{R} \mathbb{F}_p] = 6B/p = \mathsf{negl}(\lambda),$$

  since $B = \mathsf{poly}(\lambda)$. Thus, with overwhelming probability over the choice of $z, r_1, \ldots, r_\ell$, the output of the experiment is 0 in this case.

- Suppose that $\mathbf{x} \notin \mathcal{L}$ and $\mathsf{ct}'_1 \mapsto f_1$ and $\mathsf{ct}'_2 \mapsto f_2$ in T where $f_2 - \hat{z} f_1 \equiv \left(\sum_{j\in[\ell]} \alpha_j \hat{q}_j\right) + \beta$. Consider the value of $f_2 - \hat{z} f_1$ instantiated with the values of $z, r_1, \ldots, r_\ell, q_1, \ldots, q_\ell$:

$$f_2(z, r_1, \ldots, r_\ell, q_1, \ldots, q_\ell) - z \cdot f_1(z, r_1, \ldots, r_\ell, q_1, \ldots, q_\ell) = \sum_{j\in[\ell]} \alpha_j q_j + \beta = \mathbf{q}^\mathsf{T} \boldsymbol{\alpha} + \beta,$$

  where $\boldsymbol{\alpha} \in \mathbb{F}_p^\ell$ is the vector of $\alpha_1, \ldots, \alpha_\ell$. As noted above, the linear PCP query $(\mathsf{st}, \mathbf{q}_{\mathsf{inp}}, \mathbf{q})$ is sampled *after* the adversary has chosen ct'_1 and ct'_2 (and correspondingly, after it has committed to the vector $\boldsymbol{\alpha}$ and the offset β). Since $\mathbf{x} \notin \mathcal{L}$, we can appeal to soundness of Π_{LPCP} and conclude

$$\Pr[\mathcal{V}_{\mathsf{LPCP}}(\mathsf{st}, \mathbf{q}_{\mathsf{inp}}^\mathsf{T} \mathbf{x}, \mathbf{q}^\mathsf{T} \boldsymbol{\alpha} + \beta) = 1 \mid (\mathsf{st}, \mathbf{q}_{\mathsf{inp}}, \mathbf{q}) \leftarrow \mathcal{Q}_{\mathsf{LPCP}}] \leq \varepsilon.$$

  Thus, in this case, the output of the experiment is 1 with probability at most ε.

Thus, in Hyb_2, the output of the experiment is 1 with probability at most $\varepsilon + \mathsf{negl}(\lambda)$. □

The claim now follows by combining Lemmas C.11 to C.13. □

D Proof of Lemma 4.2 (Hardness of GapMWSP)

Lemma 4.2 follows by a direct adaptation of [HKLT19, Theorem 2.1], generalized to arbitrary finite fields. We give a reduction from the NP-hard problem GapLabelCover to GapMWSP. We begin by reviewing the definition and the NP-hardness of the LabelCover problem:

Definition D.1 (Gap Label Covering Problem (GapLabelCover) [Raz95, Hås01, HKLT19])**.** A LabelCover instance consists of a regular bipartite multi-graph $G = (L, R, E)$ and two finite sets Σ_L, Σ_R where $|\Sigma_L| \geq |\Sigma_R|$. Every vertex in L should be assigned a label from Σ_L and every vertex in R should be assigned a

label in Σ_R. For each edge $e \in E$ there is a projection $\pi_e \colon \Sigma_L \to \Sigma_R$. Given a labeling to the vertices of the graph (i.e. functions $\phi_L \colon L \to \Sigma_L$ and $\phi_R \colon R \to \Sigma_R$), an edge $e = (a, b) \in E$ is said to be "satisfied" if $\pi_e(\phi_L(a)) = \phi_R(b)$. For $1 \geq c > s > 0$, $\mathsf{GapLabelCover}_{c,s}$ is the promise problem of distinguishing whether the LabelCover instance is at least c-satisfiable (i.e., a c-fraction of the edges can be satisfied by some labeling) or at most s-satisfiable (i.e., every labeling can only satisfy at most an s-fraction of the edges).

Theorem D.2 (NP-Hardness of GapLabelCover [MR10, DS14])**.** *For any constant $c > 0$ and $\delta = 1/\log^c n$, $\mathsf{GapLabelCover}_{1,\delta}$ is NP-hard when the LabelCover instance satisfies $|\Sigma_L|, |\Sigma_R| \leq |L| + |R|$.*

To show NP-hardness of the GapMWSP problem, we start by showing that without loss of generality, we can always assume that $|L| > |R|$ in $\mathsf{GapLabelCover}_{1,\delta}$.

Lemma D.3. *Let $\mathcal{I}$ be a LabelCover instance over a regular bipartite multi-graph $G = (L, R, E)$. Then, there is an efficient algorithm that constructs a new LabelCover instance $\mathcal{I}'$ over a bipartite multi-graph $G' = (L', R, E')$ such that $|L'| = 2|L|$, every node in L' has equal degree, and for any $0 \leq \delta \leq 1$, instance $\mathcal{I}$ is δ-satisfiable if and only if $\mathcal{I}'$ is δ-satisfiable.*

Proof. Let $\mathcal{I} = (G = (L, R, E), \Sigma_L, \Sigma_R, \{\pi_e\}_{e \in E})$ be a LabelCover instance. Write $L = \{u_1, \ldots, u_k\}$ and $R = \{v_1, \ldots, v_\ell\}$. We construct a LabelCover instance $\mathcal{I}' = (G' = (L', R, E'), \Sigma_L, \Sigma_R, \{\pi'_e\}_{e \in E'})$ as follows:

- Let $L_1 = L$ and $L_2 = \{u'_1, \ldots, u'_k\}$ and set $L' = L_1 \cup L_2$.
- Let $E_1 = E$ and $E_2 = \{(u'_i, v_j) : (u_i, v_j) \in E\}$ and set $E' = E_1 \cup E_2$;
- For $(u_i, v_j) \in E$, set $\pi'_{u_i,v_j}, \pi'_{u'_i,v_j} \leftarrow \pi_{u_i,v_j}$.

By construction, $|L'| = 2|L|$ and G' is bipartite. Since G is regular, every node in L has the same degree; the same extends to L' by construction. It suffices to show that $\mathcal{I}$ is δ-satisfiable if and only if $\mathcal{I}'$ is δ-satisfiable:

- Suppose $\mathcal{I}$ is δ-satisfiable. Then, there exists a labeling function (ϕ_L, ϕ_R) that satisfies a δ-fraction of the edges $e \in E$. We define $\phi_{L'} \colon L' \to \Sigma_L$ to be the mapping $u_i, u'_i \mapsto \phi_L(u_i)$. By construction, this satisfies a δ fraction of the edges in E_1 and E_2.
- Suppose $\mathcal{I}'$ is δ-satisfiable. Then, there exists a labeling function $(\phi_{L'}, \phi_R)$ that satisfies a δ-fraction of the edges in $e \in E'$. Let $\phi_1 \colon L_1 \to \Sigma_L$ denote the action of $\phi_{L'}$ restricted to L_1 and let $\phi_2 \colon L_2 \to \Sigma_R$ denote the action of $\phi_{L'}$ restricted to L_2. Since $E' = E_1 \cup E_2$ and $|E_1| = |E_2|$, by construction of E_1, E_2, it must be the case that either ϕ_1 satisfies a δ-fraction of the edges in E_1 or ϕ_2 satisfies a δ-fraction of the edges in E_2. If ϕ_1 satisfies a δ-fraction of the edges in $E_1 = E$, then $\mathcal{I}$ is δ-satisfiable. If ϕ_2 satisfies a δ-fraction of the edges in E_2, then by construction of L_2 and E_2, the labeling function $\phi_L(u_i) \coloneqq \phi_2(u'_i)$ will satisfy a δ-fraction of the constraints in $E_1 = E$. □

Take any constant $c > 0$ and let $\beta = \log^c n$. Take any finite field $\mathbb{F}$ where $\log|\mathbb{F}| = \mathsf{poly}(n)$. Let $\beta' = \log^{c'} n/2$ for constant $c' > c$ such that $\beta' > \beta$. We show NP-hardness of $\mathsf{GapMWSP}_{\beta'}$ via a reduction from $\mathsf{GapLabelCover}_{1,\delta}$ for $\delta = 2/\beta' = 1/\log^{c'} n$. Since $\beta' > \beta$, this shows NP-hardness of $\mathsf{GapMWSP}_{\beta}$. Let $\mathcal{I} = (G = (L, R, E), \Sigma_L, \Sigma_R, \{\pi_e\}_{e \in E})$ be a $\mathsf{GapLabelCover}_{1,\delta}$ instance. First, by iterative application of Lemma D.3 (at most $\log|R|$ times), we can assume that $|L| > |R|$. We construct our $\mathsf{GapMWSP}_{\beta'}$ instance as follows:

- For each vertex $v \in L \cup R$ and for each possible label $\ell \in \Sigma_L \cup \Sigma_R$ for node v, introduce a variable $x_{v,\ell}$. Let $n = |L||\Sigma_L| + |R||\Sigma_R|$ be the number of variables.
- Define the following family of linear constraints over the variables $x_{v,\ell} \in \mathbb{F}$:
  - $\forall s \in L : \sum_{\ell \in \Sigma_L} x_{s,\ell} = 1$.
  - $\forall t \in R : \sum_{\ell \in \Sigma_R} x_{t,\ell} = 1$.
  - $\forall (s,t) \in E, \forall \ell \in \Sigma_R : \sum_{r \in \Sigma_L : \pi_{s,t}(r) = \ell} x_{s,r} = x_{t,\ell}$.

Let $m = |L| + |R| + |E||\Sigma_R|$ be the number of constraints. Define a matrix $\mathbf{A} \in \mathbb{F}^{m \times n}$ and a target vector $\mathbf{b} \in \mathbb{F}^m$ corresponding to the above system of linear constraints.

- We set the weight to $d = |L| + |R|$.

The $\mathsf{GapMWSP}_{\beta'}$ instance is then the tuple $\mathcal{I}' = (\mathbf{A}, \mathbf{b}, d)$. To conclude the proof, it suffices to show that there is a one-to-one correspondence between YES instances and NO instances for $\mathsf{GapLabelCover}_{1,\delta}$ and $\mathsf{GapMWSP}_{\beta'}$:

- **Low-weight solution if $\mathcal{I}$ is satisfiable:** Suppose the $\mathsf{GapLabelCover}_{1,\delta}$ instance $\mathcal{I}$ is satisfiable. Let ϕ_L, ϕ_R be the satisfying labeling function. By construction, we can set $x_{s,\phi_L(s)} = 1$ for all $s \in L$ and $x_{t,\phi_R(t)} = 1$ for all $t \in R$, and all other $x_{v,\ell} = 0$ to obtain a satisfying assignment for $\mathcal{I}'$. Moreover, there are only $|L| + |R| = d$ nonzero entries in L.
- **No low-weight solutions if $\mathcal{I}$ is not δ-satisfiable**: Suppose that $\mathcal{I}$ is not δ-satisfiable. Take any solution $\mathbf{x} \in \mathbb{F}^n$ where $\mathbf{Ax} = \mathbf{b}$. The first two constraints in the linear system ensure that for every vertex $v \in L \cup R$, there is at least one label $\ell \in \Sigma_L \cup \Sigma_R$ such that $x_{v,\ell} \neq 0$. Therefore, for every vertex v we can define a *non-empty* set of potential labels $L_v = \{\ell \in \Sigma_L \cup \Sigma_R : x_{v,\ell} \neq 0\}$.

  Consider the randomized strategy for the $\mathsf{LabelCover}$ problem where every vertex $v \in L \cup R$ is labeled with a random label $\ell_v \xleftarrow{R} L_v$. Take any edge $(s,t) \in E$, and let $\ell_t \in \Sigma_R$ be the label assigned to $t \in R$. By assumption, $x_{t,\ell_t} \neq 0$. By the third set of constraints, $\sum_{r \in \Sigma_L : \pi_{s,t}(r) = \ell_s} x_{s,r} = x_{t,\ell_t} \neq 0$. Thus, there must exist some $r \in \Sigma_L$ such that $\pi_{s,t}(r) = \ell_s$ and $x_{s,r} \neq 0$. In other words, $r \in L_s$. Since $\ell_s \xleftarrow{R} L_s$, this means that edge (s,t) is satisfied with probability at least $1/|L_s|$. Since every node in L has equal degree, the expected fraction of constraints satisfied by this labeling strategy is

$$\mathbb{E}_{s \in L}\left[\frac{1}{|L_s|}\right] \geq \frac{1}{\mathbb{E}_{s \in L}[|L_s|]},$$

  where the inequality follows from Jensen's inequality. Since $\mathcal{I}$ is not δ-satisfiable, at most a δ-fraction of the constraints are satisfiable, so this means that

$$\delta \geq \mathbb{E}_{s \in L}\left[\frac{1}{|L_s|}\right] \geq \frac{1}{\mathbb{E}_{s \in L}[|L_s|]},$$

  or equivalently, $\mathbb{E}_{s \in L}[|L_s|] \geq 1/\delta$. This means that there are at least $|L|/\delta$ variables $x_{s,\ell}$ where $s \in |L|$ and $\ell \in \Sigma_L$ that are nonzero. Since $|L| \geq |R|$ by assumption, there are at least

$$\frac{1}{\delta}|L| = \frac{1}{2\delta}(|L| + |L|) > \frac{1}{2\delta}(|L| + |R|) = \beta' d$$

  elements in $\mathbf{x}$ that are nonzero. We conclude that if $\mathcal{I}$ is not δ-satisfiable, then there are no solutions to $\mathcal{I}'$ with weight less than $\beta' d$.

Finally, we need to show that there is a Karp-Levin reduction from SAT to $\mathsf{GapMWSP}_{\beta}$. First, we use the fact that the proof of Theorem D.2 from [MR10, DS14] gives a Karp-Levin reduction from SAT to the $\mathsf{GapLabelCover}_{1,\delta}$ problem (via the PCP theorem). Moreover, our reduction above gives a Karp-Levin reduction from $\mathsf{GapLabelCover}_{1,\delta}$ to $\mathsf{GapMWSP}_{\beta'}$ (for any choice of finite field $\mathbb{F}$ where $\log|\mathbb{F}| = \mathsf{poly}(n)$). Composing the two reductions yields a Karp-Levin reduction from SAT to $\mathsf{GapMWSP}_{\beta'}$ (and correspondingly, to $\mathsf{GapMWSP}_{\beta}$) over any finite field $\mathbb{F}$ here $\log|\mathbb{F}| = \mathsf{poly}(n)$. □

E Witness Encryption from Predictable Arguments

In this section, we show how to obtain a witness encryption scheme from any 2-message predictable argument. Our construction largely follows the corresponding construction of Faonio et al. [FNV17], except we additionally consider schemes with imperfect completeness.

Construction E.1 (Witness Encryption from Predictable Argument [FNV17, adapted])**.** Let $\Pi_{\mathsf{LA}} = (\mathcal{Q}_{\mathsf{LA}}, \mathcal{P}_{\mathsf{LA}}, \mathcal{V}_{\mathsf{LA}})$ be a predictable 2-message laconic argument for an NP relation $\mathcal{R}$. Suppose the prover's message can be expressed as an ℓ-bit string (i.e., elements of $\{0,1\}^\ell$). We construct a witness encryption scheme $\Pi_{\mathsf{WE}} = (\mathsf{Encrypt}, \mathsf{Decrypt})$ for $\mathcal{R}$ as follows:

- $\mathsf{Encrypt}(1^\lambda, x, m)$ On input the security parameter λ, a statement x and a message bit $m \in \{0,1\}$, the encryption algorithm samples a random $r \xleftarrow{R} \{0,1\}^\ell$ and runs $(q, \mathsf{st}) \leftarrow \mathcal{Q}_{\mathsf{LA}}(1^\lambda, x)$, where $\mathsf{st} \in \{0,1\}^\ell$. Compute $\mathsf{ct} = (q, r, m \oplus \langle r, \mathsf{st}\rangle)$. Here, we write $\langle r, \mathsf{st}\rangle$ to denote the inner product between $r, \mathsf{st} \in \{0,1\}^\ell$ (viewed as vectors in $\mathbb{F}_2^\ell$).
- $\mathsf{Decrypt}(w, \mathsf{ct})$: On input a ciphertext $\mathsf{ct} = (q, r, m')$ and a witness w, the decryption algorithm computes $\pi \leftarrow \mathcal{P}_{\mathsf{LA}}(q, x, w) \in \{0,1\}^\ell$ and outputs $\langle r, \pi\rangle \oplus m'$.

Theorem E.2 (Witness Encryption from Predictable Arguments)**.** *If Π_{LA} is a predictable 2-message argument for $\mathcal{R}$ with completeness error c and soundness error $\varepsilon = \mathsf{negl}(\lambda)$, then Π_{WE} from Construction E.1 is a witness encryption scheme for $\mathcal{R}$ with correctness error $c/2$.*

Proof. We show correctness and semantic security.

- **Correctness:** Take any $(x, w) \in \mathcal{R}$ and $m \in \{0,1\}$. Let $\mathsf{ct} = (q, r, m') \leftarrow \mathsf{Encrypt}(1^\lambda, x, m)$. In this case, $(q, \mathsf{st}) \leftarrow \mathcal{Q}_{\mathsf{LA}}(1^\lambda, x)$, $r \xleftarrow{R} \{0,1\}^\ell$, and $m' = m \oplus \langle r, \mathsf{st}\rangle$. Consider $\mathsf{Decrypt}(\mathsf{ct}, w)$. The decryption algorithm $\mathsf{Decrypt}(w, \mathsf{ct})$ begins by computing $\pi \leftarrow \mathcal{P}_{\mathsf{LA}}(q, x, w)$. We consider two possibilities:
  - By completeness of Π_{LA}, with probability $1 - c$, $\pi = \mathsf{st}$ (since Π_{LA} is predictable). In this case, the decryption algorithm correctly outputs m.
  - With probability c, $\pi \neq \mathsf{st}$. Moreover, in this case, the value of π is computed independently of r (namely, π depends only on q, x, w while r was sampled uniformly at random from $\{0,1\}^\ell$). By pairwise independence, if $\pi \neq \mathsf{st}$,
  $$\Pr[\langle r, \pi\rangle = \langle r, \mathsf{st}\rangle \mid r \xleftarrow{R} \{0,1\}^\ell] = 1/2.$$
  Thus, the decryption algorithm outputs m with probability $1/2$ in this case.

  Thus, we see that
  $$\Pr[\mathsf{Decrypt}(w, \mathsf{ct}) = m \mid \mathsf{ct} \leftarrow \mathsf{Encrypt}(1^\lambda, x, m)] = 1 - c + \frac{c}{2} = 1 - \frac{c}{2}.$$
- **Semantic security:** Our analysis will rely on the classic Goldreich-Levin theorem [GL89], a version of which we state below:

  Theorem E.3 (Goldreich-Levin)**.** *Take any $\varepsilon > 0$. Fix some $x \in \{0,1\}^n$ and let $\mathcal{A}_x$ be an efficient (and possibly randomized) algorithm where $\Pr[\mathcal{A}_x(r) = \langle r, x\rangle \mid r \xleftarrow{R} \{0,1\}^n] \geq 1/2 + \varepsilon$. There exists a $\mathsf{poly}(n, 1/\varepsilon)$-time decoding algorithm $\mathcal{D}^{\mathcal{A}_x(\cdot)}$ that given oracle access to $\mathcal{A}_x$, outputs a list $L \subseteq \{0,1\}^n$ such that $|L| = \mathsf{poly}(n, 1/\varepsilon)$ and $x \in L$ with probability at least $1/2$.*

  Turning back to semantic security, suppose that there exists an efficient adversary $\mathcal{A}$ and some $x \notin \mathcal{L}$ such that
  $$\Pr[\mathcal{A}(1^\lambda, \mathsf{ct}_b) = b \mid b \xleftarrow{R} \{0,1\}, \mathsf{ct}_b \leftarrow \mathsf{Encrypt}(1^\lambda, x, b)] \geq \frac{1}{2} + \varepsilon.$$
  for some $\varepsilon = 1/\mathsf{poly}(\lambda)$. Let t be a bound on the number of bits of randomness needed by $\mathcal{Q}_{\mathsf{LA}}$. Namely, to sample $(q, \mathsf{st}) \leftarrow \mathcal{Q}_{\mathsf{LA}}(1^\lambda, x)$, one first samples $\rho \xleftarrow{R} \{0,1\}^t$ and then computes $(q, \mathsf{st}) \leftarrow \mathcal{Q}_{\mathsf{LA}}(1^\lambda, x; \rho)$. This means that
  $$\Pr\left[\mathcal{A}(1^\lambda, (q, r, b \oplus \langle r, \mathsf{st}\rangle)) = b \;\middle|\; \begin{array}{c} b \xleftarrow{R} \{0,1\}, r \xleftarrow{R} \{0,1\}^\ell, \rho \xleftarrow{R} \{0,1\}^t; \\ (q, \mathsf{st}) \leftarrow \mathcal{Q}_{\mathsf{LA}}(1^\lambda, x; \rho) \end{array}\right] \geq \frac{1}{2} + \varepsilon.$$
  We use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ that on input (q, r) predicts the value of $\langle r, \mathsf{st}\rangle$, where $(q, \mathsf{st}) \leftarrow \mathcal{Q}_{\mathsf{LA}}(1^\lambda, x)$ and $r \xleftarrow{R} \{0,1\}^\ell$:

– On input (q, r), algorithm $\mathcal{B}$ samples $b \xleftarrow{R} \{0,1\}$ and invokes $\mathcal{A}$ on input (q, r, b). If $\mathcal{A}$ outputs b, then $\mathcal{B}$ outputs 0. Otherwise $\mathcal{B}$ outputs 1.

It is easy to see that by construction,

$$\Pr\left[\mathcal{B}(1^\lambda, (q, r)) = \langle r, \mathsf{st}\rangle \;\middle|\; \begin{array}{c} r \xleftarrow{R} \{0,1\}^\ell, \rho \xleftarrow{R} \{0,1\}^t; \\ (q, \mathsf{st}) \leftarrow \mathcal{Q}_{\mathsf{LA}}(1^\lambda, x; \rho) \end{array}\right] \geq \frac{1}{2} + \varepsilon.$$

We use $\mathcal{B}$ to construct an adversary $\mathcal{B}'$ that breaks soundness of Π_{LA} (for the instance $x \notin \mathcal{L}$):

1. On input the security parameter λ, the instance x, and the query q, run the Goldreich-Levin list-decoding algorithm $\mathcal{D}^{\mathcal{B}(1^\lambda, (q, \cdot))}$ (Theorem E.3) where the oracle queries are implemented as follows:
   – On input $r \in \{0,1\}^\ell$, output $\mathcal{B}(1^\lambda, (q, r))$.

   Let $L \subseteq \{0,1\}^\ell$ be the list output by the list-decoding algorithm.
2. Output $\pi \xleftarrow{R} L$.

We now analyze the success probability of algorithm $\mathcal{B}'$. First, for a fixed element $\rho \in \{0,1\}^t$ and the associated query $(q, \mathsf{st}) \leftarrow \mathcal{Q}_{\mathsf{LA}}(1^\lambda, x; \rho)$, define the algorithm $\mathcal{B}_\rho(r)$ to be the algorithm that on input $r \in \{0,1\}^\ell$ outputs $\mathcal{B}(1^\lambda, (q, r))$. Let $S \subseteq \{0,1\}^t$ be the set of elements $\rho \in \{0,1\}^t$ such that

$$\Pr[\mathcal{B}_\rho(r) = \langle r, \mathsf{st}\rangle \mid r \xleftarrow{R} \{0,1\}^\ell, (q, \mathsf{st}) \leftarrow \mathcal{Q}_{\mathsf{LA}}(1^\lambda, x; \rho)] \geq \frac{1}{2} + \frac{\varepsilon}{2}. \tag{E.1}$$

By an averaging argument,

$$\Pr[\rho \in S \mid \rho \xleftarrow{R} \{0,1\}^t] \geq \varepsilon.$$

In the soundness game, the challenger begins by sampling $\rho \xleftarrow{R} \{0,1\}^t$ and $(q, \mathsf{st}) \leftarrow \mathcal{Q}_{\mathsf{LA}}(1^\lambda, x; \rho)$, and algorithm $\mathcal{B}'$ is given as input $(1^\lambda, x, q)$. Thus, with probability ε, the randomness ρ sampled by the challenger lies in the set S. In this case, the oracle $\mathcal{B}(1^\lambda, (q, \cdot)) \equiv \mathcal{B}_\rho(\cdot)$ in $\mathcal{B}'$ satisfies Eq. (E.1), so we can appeal to the Goldreich-Levin theorem (Theorem E.3) to conclude that the list L output by $\mathcal{D}^{\mathcal{B}(1^\lambda, (q, \cdot))}$ contains st with probability at least $1/2$. If $\mathsf{st} \in L$, then $\mathcal{B}'$ outputs st with probability $1/|L|$, and since Π_{LA} is predictable, $\mathcal{V}_{\mathsf{LA}}(\mathsf{st}, \mathsf{st}) = 1$. Thus, $\mathcal{B}'$ breaks soundness with probability

$$\Pr[\mathcal{B}'(1^\lambda, x, q) = \mathsf{st} \mid (q, \mathsf{st}) \leftarrow \mathcal{Q}_{\mathsf{LA}}(1^\lambda, x)] \geq \varepsilon \cdot \frac{1}{2} \cdot \frac{1}{|L|} = \frac{1}{\mathsf{poly}(\ell, 1/\varepsilon)} = \frac{1}{\mathsf{poly}(\lambda)},$$

since $\ell = \mathsf{poly}(\lambda)$ and $\varepsilon = 1/\mathsf{poly}(\lambda)$ by assumption. Moreover, the running time of $\mathcal{B}'$ is bounded by the product of the running times of the list-decoding algorithm and the running time of $\mathcal{B}$, both of which are $\mathsf{poly}(\lambda, \ell, 1/\varepsilon) = \mathsf{poly}(\lambda)$. Thus, $\mathcal{B}'$ breaks soundness with non-negligible probability and the claim holds. □