# A Hidden-Bits Approach to Statistical ZAPs from LWE

Eli Bradley[1], George Lu[1], Shafik Nassar[1], Brent Waters[1,2], and David J. Wu[1]

[1]UT Austin
[2]NTT Research

**Abstract**

We give a new approach for constructing statistical ZAP arguments (a two-message public-coin statistically witness indistinguishable argument) from quasi-polynomial hardness of the learning with errors (LWE) assumption with a polynomial modulus-to-noise ratio. Previously, all ZAP arguments from lattice-based assumptions relied on correlation-intractable hash functions. In this work, we present the first construction of a ZAP from LWE via the classic hidden-bits paradigm. Our construction matches previous lattice-based schemes by being public-coin and satisfying statistical witness indistinguishability.

## 1 Introduction

Zero-knowledge proofs for NP [GMR85] allow a prover to convince a verifier that an NP statement is true without revealing anything more than the fact that the statement is true. However, achieving zero-knowledge requires either a trusted setup (i.e., a common reference string) or interaction. Many works have studied the round complexity of zero-knowledge proofs for NP in the plain model, and it is known, for instance, that two-round zero-knowledge proofs for NP are impossible in the plain model [GO94, BLV03] under standard worst-case complexity assumptions (e.g., that $\mathsf{NP} \not\subseteq \mathsf{BPP}$). One way to overcome the barrier of two-round zero-knowledge in the plain model is to relax zero-knowledge to witness indistinguishability [FS90], which asks that a proof constructed using an NP witness $w_0$ is indistinguishable from a proof constructed using a witness $w_1$.

**ZAPs.** Dwork and Naor [DN00] gave the first construction of a two-round public-coin witness-indistinguishable proof (i.e., "ZAPs") in the plain model, and moreover, showed how to construct it from any non-interactive zero-knowledge (NIZK) proof in the uniform random string model (e.g., [FLS90]). Critically, the [DN00] approach relied on both *statistical* soundness and that the common reference string is a *uniform* random string. In the context of the ZAP, the public-coin property means the verifier's first message is a uniform random string (i.e., the random coins of the verifier). As such, ZAPs are publicly-verifiable since anyone who sees the verifier's first message can run the verification algorithm. Following the work of Dwork and Naor, there have been many constructions of ZAPs from pairing-based assumptions [GOS06a, CKSU21], lattice-based assumptions [LVW19, BFJ+20, GJJM20], group-based assumptions (without pairings) [JJ21, CKSU21], and from indistinguishability obfuscation (together with one-way functions) [BP15]. Several works also considered a relaxation to ZAPs with private randomness [KKS18, LVW20].

**This work: statistical ZAP arguments.** In this work, we focus on *statistical ZAP arguments* where witness indistinguishability holds against computationally-unbounded adversaries (and soundness holds against computationally-bounded adversaries). Public-coin statistical ZAP arguments are currently known from the decisional Diffie-Hellman (DDH) assumption in a pairing-free group [JJ21] as well as from the plain learning with errors (LWE) assumptions [LVW19, BFJ+20, GJJM20]. Notably, *all* of these approaches start by constructing a correlation-intractable hash function and using it to instantiate the Fiat-Shamir transform [CCRR18, CCH+19]. In this work, we ask whether we

can construct a ZAP from LWE *without* relying on correlation intractability. Specifically, motivated by the recent advances in realizing NIZKs from LWE via the classic hidden-bits paradigm due to Waters [Wat24] and recently improved upon by Waters, Wee, and Wu [WWW25], we ask whether there is a hidden-bits model approach for building ZAPs from LWE. Indeed, the very first ZAP by [DN00] showed how to leverage the hidden-bits model NIZK from [FLS90] to obtain a computational ZAP proof. Later, [CKSU21] showed how to leverage the pairing-based hidden-bits model NIZK from [LPWW20] to also obtain a public-coin statistical ZAP argument using bilinear maps. In this work, we show how to construct a statistical ZAP argument from LWE via the hidden-bits approach. Our main result can be stated as follows:

**Theorem 1.1** (Informal)**.** *Assuming quasi-polynomial hardness of* LWE *with a polynomial modulus-to-noise ratio, there exists a statistical ZAP argument for* NP *with non-adaptive soundness.*

If we are willing to assume sub-exponential hardness of LWE, then we obtain an adaptively-sound statistical ZAP for statements of a priori bounded length. We refer to Remark 3.3 for additional details.

**Comparison and discussion.** Theorem 1.1 achieves the same properties as previous lattice-based ZAPs [LVW19, BFJ+20, GJJM20], which also relied on quasi-polynomial security of LWE (with polynomial modulus-to-noise ratio). The previous approaches all relied on correlation-intractable hash functions. Essentially, in these constructions, both the prover and the verifier need to homomorphically evaluate the decryption function for a public-key encryption scheme (in order to evaluate the correlation-intractable hash function). In this work, we seek a conceptually-different approach that does not rely on correlation intractability.

We believe there is value in exploring different approaches for realizing a single cryptographic goal. For instance, in the setting of NIZKs, there have been two general (and incomparable) paradigms: the hidden-bits paradigm [FLS90, BY92, CHK03, GR13, CL18, LPWW20, KMY23, CW23, Wat24, BWW24, WWW25] and the correlation-intractability paradigm [CCRR18, CCH+19, PS19, CKU20, BKM20, JJ21, CJJQ23, DJJ24] that have yielded constructions from a broad array of algebraic assumptions. There are also many approaches that do not fall into either paradigm [BFM88, GOS06b, SW14, BKP+24]. Here, similar algebraic assumptions can yield very different approaches for realizing the primitive, and this enriches our understanding of the particular primitive. Moreover, having a broad set of approaches and constructions can enable new trade-offs for concrete efficiency. We believe the same is true for ZAPs. Here again, there have been two broad categories of approaches for constructing ZAPs, based either on the hidden-bits model or on the correlation-intractability approach. While previous lattice-based ZAPs relied critically on correlation-intractable hash functions, our approach shows that a construction is also feasible via the earlier hidden-bits approach.

## 1.1 Technical Overview

Our starting point for constructing a statistical ZAP argument from LWE is the recent (dual-mode) NIZK in the hidden-bits model from the work of [Wat24] (and subsequently improved upon in the work of [WWW25]). Both constructions operate by constructing a (dual-mode) *hidden-bits generator* [FLS90, QRW19, LPWW20]. We start by recalling this notion.

**Hidden-bits generators.** Roughly speaking, a hidden-bits generator (HBG) [FLS90, QRW19] is a cryptographic primitive in the common reference string model that allows the prover to generate a random string, alongside a commitment to the string and a local opening to each bit. Moreover, the scheme should satisfy the following properties:

- **Hiding:** First, we require that the openings to a subset of the bits hide the unopened bits.
- **Binding:** Second, given the commitment (and the CRS), the prover can only open to one possible hidden-bits string.

A dual mode HBG can be instantiated in one of two computationally indistinguishable modes: statistically hiding or statistically binding.

**Shifted multi-preimage sampling.** To construct a hidden-bits generator from lattice assumptions, the work of [WWW25] explicitly defined the *shifted multi-preimage sampling problem*, which was implicitly considered in the previous work of [Wat24]. The problem is parameterized by a collection of matrices $\mathbf{A}_1, \ldots, \mathbf{A}_t \in \mathbb{Z}_q^{n \times t}$ and requires the following: given target vectors $\mathbf{t}_1, \ldots, \mathbf{t}_\ell \in \mathbb{Z}_q^n$, sample a random vector $\mathbf{c} \xleftarrow{R} \mathbb{Z}_q^n$ with short discrete Gaussian preimages $\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_\ell \in \mathbb{Z}_q^t$ satisfying $\mathbf{A}_i \boldsymbol{\pi}_i = \mathbf{t}_i + \mathbf{c}$ for all $i \in [\ell]$. That is, $\boldsymbol{\pi}_i \leftarrow \mathbf{A}_i^{-1}(\mathbf{t}_i + \mathbf{c})$ for each $i \in [\ell]$, where $\mathbf{x} \leftarrow \mathbf{A}_i^{-1}(\mathbf{y})$ denotes the discrete Gaussian distribution conditioned on $\mathbf{A}_i \mathbf{x} = \mathbf{y}$. The goal is to publish a hint that allows one to efficiently sample $(\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_\ell, \mathbf{c})$, while ensuring that problems like LWE remain hard with respect to any individual matrix $\mathbf{A}_i$. In particular, this latter requirement rules out the possibility of giving a trapdoor for each individual $\mathbf{A}_i$ as the hint. The work of [WWW25] shows how to solve this problem for a specific distribution of *correlated* matrices $\mathbf{A}_1, \ldots, \mathbf{A}_t \in \mathbb{Z}_q^{n \times t}$. Moreover, although the matrices are correlated, the marginal distribution of each individual matrix is actually uniform over $\mathbb{Z}_q^{n \times t}$.

**The [Wat24, WWW25] hidden-bits generator.** We can now describe the LWE-based hidden-bits generator from [Wat24, WWW25] through the lens of the shifted multi-preimage sampling problem from [WWW25]:

- The common reference string (for the hidden-bits generator) consists of the CRS for the shifted multi-preimage trapdoor sampler together with $\ell$ vectors $\mathbf{v}_1, \ldots, \mathbf{v}_\ell \in \mathbb{Z}_q^t$. The distribution of the vectors $\mathbf{v}_i$ varies depending on whether we want the hidden-bits generator to be statistically binding or statistically hiding. The CRS for the shifted multi-preimage trapdoor sampler defines a set of $\ell$ matrices $\mathbf{A}_1, \ldots, \mathbf{A}_\ell \in \mathbb{Z}_q^{n \times t}$ and a hint td for solving the shifted multi-preimage sampling problem with respect to matrices $\mathbf{A}_1, \ldots, \mathbf{A}_\ell$.
- To generate the hidden-bits string, the user samples $(\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_\ell, \mathbf{c})$ using the multi-preimage sampler with targets $\mathbf{t}_i = \mathbf{0}$ for all $i \in [\ell]$. This means $\mathbf{A}_i \boldsymbol{\pi}_i = \mathbf{c}$ for all $i \in [\ell]$. The commitment of the hidden-bits generator is the vector $\mathbf{c} \in \mathbb{Z}_q^n$. Each bit $\rho_i$ is now defined as $\rho_i = \lfloor \mathbf{v}_i^\intercal \boldsymbol{\pi}_i \rceil \in \{0, 1\}$, where $\lfloor \cdot \rceil$ denotes a rounding operation. The hidden-bits string is defined as $\boldsymbol{\rho} = \rho_1 \cdots \rho_\ell$, and the openings are $\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_\ell$.
- In hiding mode, the vectors $\mathbf{v}_1, \ldots, \mathbf{v}_\ell$ in the CRS are uniform over $\mathbb{Z}_q^t$. The security proof of [WWW25] now relies on the min-entropy of the conditional discrete Gaussian distribution $\mathbf{A}_i^{-1}(\mathbf{c})$ (even given $\mathbf{c}$) and the randomness of $\mathbf{v}_i$ to argue that $\mathbf{v}_i^\intercal \boldsymbol{\pi}_i$ is statistically close to uniform over $\mathbb{Z}_q$ (even given $\mathbf{v}_i$, but not $\boldsymbol{\pi}_i$) using the leftover hash lemma [HILL99].
- In binding mode, the vectors $\mathbf{v}_1, \ldots, \mathbf{v}_\ell$ are LWE samples (i.e., $\mathbf{v}_i^\intercal = \mathbf{s}_i^\intercal \mathbf{A}_i + \mathbf{e}_i^\intercal$). In this case, the commitment $\mathbf{c}$ determines the bit. For simplicity, if we ignore the error $\mathbf{e}_i$ and just consider the inner product, we have $\mathbf{v}_i^\intercal \boldsymbol{\pi}_i = \mathbf{s}_i^\intercal (\mathbf{A}_i \boldsymbol{\pi}_i) = \mathbf{s}_i^\intercal \mathbf{c}$. Specifically, the LWE secret $\mathbf{s}_i$ and the commitment $\mathbf{c}$ completely determines the value of $\mathbf{v}_i^\intercal \boldsymbol{\pi}_i$, and correspondingly, the the bit $\rho_i = \lfloor \mathbf{v}_i^\intercal \boldsymbol{\pi}_i \rceil = \lfloor \mathbf{s}_i^\intercal \mathbf{c} \rceil$.
- Finally, the CRS in hiding mode is computationally indistinguishable from the CRS in binding mode by the LWE assumption.

**Problem: lack of a trusted setup.** While the hidden-bits generator immediately implies a NIZK for NP [FLS90, QRW19], the same is not true for ZAPs. This is because a hidden-bits generator requires a common reference string, and the assumption is that the CRS is sampled *honestly*. In a ZAP, we do not have any trusted setup, so it is not clear where this common reference string comes from. Certainly, neither the prover nor the verifier should unilaterally choose the CRS, as this would violate either soundness or witness indistinguishability, respectively.

**Relying on re-randomization.** The work of Dwork and Naor [DN00] shows that NIZKs can nonetheless be used to obtain ZAPs by having the prover and the verifier *jointly* sample a CRS via the following general template (also used in many subsequent works [LVW19, BFJ+20, GJJM20, CKSU21]):

- The verifier picks an initial common reference string $\mathsf{crs}_V$, but then the prover re-randomizes it by choosing a tweak $\mathsf{crs}_P$.
- The prover cannot "completely" re-randomize the CRS, since this would give the prover the ability to choose the full CRS. Instead, there should be a small set $S$ of "tweaks" that the prover can choose from.

- The prover communicates the tweak $\mathsf{crs}_P$ it used to the verifier. The proof is then constructed with respect to the CRS derived from $(\mathsf{crs}_V, \mathsf{crs}_P)$.

For example, suppose the verifier chooses $\mathsf{crs}_V$ as its initial CRS. Then the prover may choose a string $\mathsf{crs}_P \in S$ from the (small) set $S$, and re-randomize the CRS by computing $\mathsf{crs} = \mathsf{crs}_P \oplus \mathsf{crs}_V$. The prover then generates a proof using the re-randomized reference string crs.

**Arguing soundness via complexity leveraging.** To prove soundness, we use a complexity-leveraging idea similar to that from [LVW19, BFJ+20, GJJM20, CKSU21] where the verifier "guesses" the re-randomization $\mathsf{crs}_P$ that the prover will use. Namely, the verifier first samples a common reference string $\mathsf{crs}_{\mathsf{bind}}$ for which the hidden-bits generator is binding, and then chooses $\mathsf{crs}_V = \mathsf{crs}_{\mathsf{bind}} \oplus \mathsf{crs}_P$ as its common reference string. If the verifier correctly guesses the prover's common reference string $\mathsf{crs}_P$, then the prover will compute $\mathsf{crs} = \mathsf{crs}_V \oplus \mathsf{crs}_P = \mathsf{crs}_{\mathsf{bind}}$ and generate a proof with respect to $\mathsf{crs}_{\mathsf{bind}}$. Since the hidden-bits generator is binding with respect to $\mathsf{crs}_{\mathsf{bind}}$, this guarantees soundness for the overall ZAP. If there are $|S|$ possibilities for $\mathsf{crs}_P$, then the verifier correctly guesses $\mathsf{crs}_P$ with probability $1/|S|$. To show this works, we require the two following properties hold:

- The reference string $\mathsf{crs}_{\mathsf{bind}}$ that the verifier samples should hide $\mathsf{crs}_P$. This is necessary to ensure that the prover does not "avoid" choosing the tweak the verifier guessed. More precisely, an efficient prover should not be able to distinguish a reference string $\mathsf{crs}_V = \mathsf{crs}_{\mathsf{bind}} \oplus \mathsf{crs}_P$ from $\mathsf{crs}'_V = \mathsf{crs}_{\mathsf{bind}} \oplus \mathsf{crs}'_P$ for any (adversarially-chosen) pair of tweaks $\mathsf{crs}_P, \mathsf{crs}'_P$, with probability better than $\varepsilon_{\mathsf{dis}}$. In this case, if the original prover is able to break the binding property of the hidden bits generator with probability $\varepsilon$, then with probability $\varepsilon/|S| - \varepsilon_{\mathsf{dis}}$, the prover will choose $\mathsf{crs}_P$ as its tweak and break binding with respect to the tweaked CRS $\mathsf{crs}_V \oplus \mathsf{crs}_P = \mathsf{crs}_{\mathsf{bind}}$.
- Next, we require that the probability $\varepsilon_{\mathsf{bind}}$ that an adversary breaks binding with respect to a reference string $\mathsf{crs}_{\mathsf{bind}}$ sampled in binding mode to be small.

If both properties hold, then $\varepsilon/|S| - \varepsilon_{\mathsf{dis}} \leq \varepsilon_{\mathsf{bind}}$, or equivalently, $\varepsilon \leq |S| \cdot (\varepsilon_{\mathsf{dis}} + \varepsilon_{\mathsf{bind}})$. In the context of the [WWW25] construction, these two requirements are satisfied as follows:

- In [WWW25], the reference string $\mathsf{crs}_{\mathsf{bind}}$ in binding mode is pseudorandom under the LWE assumption. An adversary that can distinguish between $\mathsf{crs}_{\mathsf{bind}} \oplus \mathsf{crs}_P$ and $\mathsf{crs}_{\mathsf{bind}} \oplus \mathsf{crs}'_P$ with advantage $\varepsilon_{\mathsf{bind}}$ can break LWE with the same advantage $\varepsilon_{\mathsf{bind}}$.
- Next, when the common reference string $\mathsf{crs}_{\mathsf{bind}}$ is sampled in binding mode, the resulting construction is statistically binding (e.g., the advantage $\varepsilon_{\mathsf{bind}}$ is exponentially small).

The remaining question is how big the set $S$ should be. In [WWW25], when the CRS is binding, there is a trapdoor that can be used to extract the actual hidden-bit string (which in turn would violate zero-knowledge or witness indistinguishability when used to construct a NIZK or a ZAP, respectively). Therefore, we cannot have $1/|S|$ be inverse polynomial, as this allows the verifier to break witness indistinguishability with inverse polynomial advantage (namely, the verifier would choose $\mathsf{crs}_{\mathsf{bind}}$ with knowledge of the associated trapdoor). Thus, to argue hiding, we take $|S|$ to be quasi-polynomial (i.e., $|S| = 2^{\mathsf{polylog}(\lambda)}$) and rely on quasi-polynomial hardness of LWE (i.e., the LWE distinguishing advantage of any efficient adversary is at most $\varepsilon_{\mathsf{dis}} = 1/2^{\mathsf{polylog}(\lambda)}$).

**Specialized CRS re-randomization for statistical hiding.** Arguing hiding is more complex and the set of possible tweaks $S$ must be carefully chosen based on the specific algebraic structure of the base scheme. Recall first that the CRS for the hidden-bits generator consists of a common reference string $\mathsf{crs}_{\mathsf{samp}}$ for the shifted multi-preimage trapdoor sampler and the vectors $\mathbf{v}_1, \ldots, \mathbf{v}_\ell \in \mathbb{Z}_q^t$. In our construction, the verifier first picks an initial $\mathsf{crs}_{\mathsf{samp}}$ for the shifted multi-preimage trapdoor sampler and a set of vectors $\tilde{\mathbf{v}}_1, \ldots, \tilde{\mathbf{v}}_\ell \in \mathbb{Z}_q^t$. The prover re-randomizes the set of vectors $\tilde{\mathbf{v}}_1, \ldots, \tilde{\mathbf{v}}_\ell$, but does *not* change $\mathsf{crs}_{\mathsf{samp}}$.

More concretely, let $\mathsf{crs}_V = (\mathsf{crs}_{\mathsf{samp}}, \tilde{\mathbf{v}}_1, \ldots, \tilde{\mathbf{v}}_\ell)$ be the reference string chosen by the verifier. In our approach, the prover chooses a small subset of indices $I \subseteq [t]$ of the vectors and *fully* re-randomizes the components of $\tilde{\mathbf{v}}_i$ at these indices. Specifically, the prover chooses a single random *sparse* vector $\boldsymbol{\delta} \in \mathbb{Z}_q^t$, and then shifts each of the vectors $\tilde{\mathbf{v}}_i$ by $\boldsymbol{\delta}$ by computing $\mathbf{v}_i = \tilde{\mathbf{v}}_i + \boldsymbol{\delta}$. The prover then uses $\mathsf{crs} = (\mathsf{crs}_{\mathsf{samp}}, \mathbf{v}_1, \ldots, \mathbf{v}_\ell)$ as the CRS for the hidden-bits generator. Observe that if $\boldsymbol{\delta}$ has $N = \mathsf{polylog}(\lambda)$ non-zero entries (where $\lambda$ is a security parameter), and if $t, q = \mathsf{poly}(\lambda)$, then there are at most $t^N q^N = 2^{\mathsf{polylog}(\lambda)}$ options for $\boldsymbol{\delta}$, so the set of tweaks indeed has quasi-polynomial size.

**Introducing min-entropy into proof generation.** The hiding analysis of [WWW25] employs a min-entropy argument which relies on the *preimage distribution* property of the shifted multi-preimage trapdoor sampler. This property essentially asserts that when the CRS for the shifted multi-preimage trapdoor sampler is *honestly* generated, then the distribution of $(\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_\ell, \mathbf{c})$ output by the multi-preimage sampler is statistically close to the distribution where $\mathbf{c} \xleftarrow{R} \mathbb{Z}_q^n$ and $\boldsymbol{\pi}_i \leftarrow \mathbf{A}_i^{-1}(\mathbf{c})$. Then, by relying on min-entropy of the (conditional) discrete Gaussian distribution $\mathbf{A}_i^{-1}(\mathbf{c})$, they show how to extract a random bit (by computing $\rho_i = \lfloor \mathbf{v}_i^\mathsf{T} \boldsymbol{\pi}_i \rceil$ and appealing to the leftover hash lemma where $\mathbf{v}_i$ is the seed for the extractor).

This preimage distribution property for the shifted multi-preimage trapdoor sampler critically assumed that the CRS (and correspondingly, the matrices $\mathbf{A}_1, \ldots, \mathbf{A}_\ell$) are sampled honestly. In particular, the previous argument relies on the marginal distribution of $\mathbf{A}_i$ being uniform. Unfortunately, in our case, the CRS for the multi-preimage sampler is chosen *adversarially* by the verifier, so we can no longer rely on $\boldsymbol{\pi}_i$ being distributed according to a discrete Gaussian or even on it having high min-entropy. To overcome this problem, the prover needs to introduces additional min-entropy during proof generation via the following procedure:

- First, for each $i \in [\ell]$, the prover samples a uniform vector $\mathbf{w}_i \xleftarrow{R} \{0,1\}^t$.
- Instead of setting the shifted multi-preimage trapdoor sampler targets to be $\mathbf{t}_1 = \cdots = \mathbf{t}_\ell = \mathbf{0}$ as in [WWW25], the prover now defines the targets to be $\mathbf{t}_1 = \mathbf{A}_1\mathbf{w}_1, \ldots, \mathbf{t}_\ell = \mathbf{A}_\ell\mathbf{w}_\ell$. This yields $(\boldsymbol{\pi}'_1, \ldots, \boldsymbol{\pi}'_\ell, \mathbf{c})$ such that $\mathbf{A}_i\boldsymbol{\pi}'_i = \mathbf{t}_i + \mathbf{c} = \mathbf{A}_i\mathbf{w}_i + \mathbf{c}$ for each $i \in [\ell]$.
- The prover now defines the openings to be $\boldsymbol{\pi}_i = \boldsymbol{\pi}'_i - \mathbf{w}_i$. Observe that this still preserves the [WWW25] verification invariant: $\mathbf{A}_i\boldsymbol{\pi}_i = \mathbf{A}_i(\boldsymbol{\pi}'_i - \mathbf{w}_i) = \mathbf{t}_i + \mathbf{c} - \mathbf{t}_i = \mathbf{c}$.
- The key observation is that the additional vector $\mathbf{w}_i$ introduces additional min-entropy into $\boldsymbol{\pi}_i$. In particular, $\boldsymbol{\pi}_i$ has high min-entropy even given the commitment $\mathbf{c}$ and the other openings $\boldsymbol{\pi}_j$ for all $j \neq i$. The latter property holds because $(\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_\ell, \mathbf{c})$ are functions of $\mathbf{t}_1 = \mathbf{A}_1\mathbf{w}_1, \ldots, \mathbf{t}_\ell = \mathbf{A}_\ell\mathbf{w}_\ell$, and $\mathbf{A}_i$ is a *compressing* function. Namely, the value of $\mathbf{t}_i = \mathbf{A}_i\mathbf{w}_i$ can only leak a limited number of bits of information about $\mathbf{w}_i$.

**Preserving min-entropy.** Having established that the opening $\boldsymbol{\pi}_i$ has high min-entropy (even given $\mathbf{c}$ and $\boldsymbol{\pi}_j$ for $j \neq i$), it remains to argue that the bit $\rho_i = \lfloor \mathbf{v}_i^\mathsf{T}\boldsymbol{\pi}_i \rceil$ is statistically close to uniform. By definition, $\mathbf{v}_i = \tilde{\mathbf{v}}_i + \boldsymbol{\delta}$ so

$$\mathbf{v}_i^\mathsf{T}\boldsymbol{\pi}_i = \tilde{\mathbf{v}}_i^\mathsf{T}\boldsymbol{\pi}_i + \boldsymbol{\delta}^\mathsf{T}\boldsymbol{\pi}_i.$$

Our argument critically relies on the fact that the re-randomization term $\boldsymbol{\delta}^\mathsf{T}\boldsymbol{\pi}_i$ is statistically close to uniform over $\mathbb{Z}_q$. Recall that $\boldsymbol{\delta} \in \mathbb{Z}_q^t$ is a sparse vector with up to $N$ non-zero entries (and where the non-zero entries are uniformly random). Let $J \subseteq [t]$ be the indices where $\boldsymbol{\delta}$ is non-zero. Then $\boldsymbol{\delta}^\mathsf{T}\boldsymbol{\pi}_i = \sum_{j \in J} \delta_j \pi_{i,j}$. If we can argue that the sub-vector $(\pi_{i,j})_{j \in J}$ has high min-entropy, then we can appeal to the leftover hash lemma (treating $\boldsymbol{\delta}$ as the seed for the extractor and $(\pi_{i,j})_{j \in J}$ as the min-entropy source) and conclude that $\boldsymbol{\delta}^\mathsf{T}\boldsymbol{\pi}_i$ is statistically close to uniform over $\mathbb{Z}_q$. To do so, we use a standard entropy preservation argument (c.f., [NZ96, Vad06, ADW09]) that says that if $\boldsymbol{\pi}_i \in \mathbb{Z}_q^t$ has high min-entropy, then projecting $\boldsymbol{\pi}_i$ onto a *random* set of coordinates $J \subseteq [t]$ still yields a string with high min-entropy. By our earlier argument, the openings $\boldsymbol{\pi}_i$ have high min-entropy so projecting $\boldsymbol{\pi}_i$ onto the random subset $J \subseteq [N]$ preserves the min-entropy. We can now appeal to the leftover hash lemma to conclude that $\boldsymbol{\delta}^\mathsf{T}\boldsymbol{\pi}_i$ is statistically close to uniform over $\mathbb{Z}_q$. Since $\mathbf{v}_i^\mathsf{T}\boldsymbol{\pi}_i = \tilde{\mathbf{v}}_i^\mathsf{T}\boldsymbol{\pi}_i + \boldsymbol{\delta}^\mathsf{T}\boldsymbol{\pi}_i$, this means that $\mathbf{v}_i^\mathsf{T}\boldsymbol{\pi}_i$ is statistically close to uniform, which completes the hiding analysis.

**Interactive hidden-bits generators.** Rather than build a ZAP directly, the work of [CKSU21] introduces an intermediate abstraction called an *interactive hidden-bits generator*, which models the key ingredients we described above. The approach described above readily maps to this abstraction. Once we have constructed an interactive hidden-bits generator from quasi-polynomial hardness of LWE, we can directly invoke the [CKSU21] compiler to obtain a public-coin statistical ZAP argument for NP. We give the formal definition of an interactive hidden-bits generator in Section 3, and give our construction in Section 4.

The previous work of [CKSU21] describe a pairing-based construction of an interactive hidden-bits generator. In their approach, the prover re-randomizes a matrix encoded in the exponent by applying a perturbation to the

diagonal entries. They show that with overwhelming probability, this yields a full-rank matrix in the exponent, which ensures hiding for the unopened bits of the hidden-bits string. The starting point of our approach is the hidden-bits generator from [Wat24, WWW25], which has a different and incomparable structure. Correspondingly, our lattice-based approach uses an entirely different re-randomization approach.

**Concurrent work.** Very recently, a concurrent work by [BCD+25] also followed up on the work of [Wat24] and presents a NIZK from LWE with a polynomial modulus (similar to [WWW25]). Like [WWW25], they do not construct a ZAP, although we believe a variation of our techniques can also be applied to their scheme as well. However, such an adaptation would inherit properties that result in a less efficient scheme (e.g., the verifier message would be $\ell^2$, where $\ell$ is the output length of the hidden-bits generator and the size of the openings would be linear in $\ell$). Their work also introduces a new statistically-sound NIZK from DDH and LPN. It is an interesting open question whether our techniques can be helpful to obtain a ZAP from DDH and LPN.

## 2 Preliminaries

Throughout this work, we write $\lambda$ to denote the security parameter. We write $\boldsymbol{\eta}_i$ to denote the $i^{\text{th}}$ standard basis vector (with dimension implicit from the context). We say that a function $\mu(\lambda)$ is negligible if $\mu(\lambda) = o(\lambda^{-c})$ for every constant $c > 0$. We denote this by writing $\mu(\lambda) = \mathsf{negl}(\lambda)$. We say that an event happens with overwhelming probability if its complement happens with negligible probability. For any positive integer $n \in \mathbb{N}$, we write $[n] \coloneqq \{1, \ldots, n\}$. For any positive integer $q \in \mathbb{N}$, we write $\mathbb{Z}_q$ to denote the ring of integers modulo $q$. Throughout this work, we use bold uppercase letters (e.g., $\mathbf{A}, \mathbf{B}$) to denote matrices and bold lowercase letters (e.g., $\mathbf{u}, \mathbf{v}$) to denote vectors. For a vector $\mathbf{v} = (v_1, \ldots, v_n) \in \mathbb{Z}_q^n$, and any vector of indices $I = (i_1 \ldots, i_k) \in [n]^k$, we write $\mathbf{v}[I]$ to denote the projection of $\mathbf{v}$ on $I$, that is $\mathbf{v}[I] = (v_{i_1}, \ldots, v_{i_k})$. We say that an algorithm is efficient if it runs in probabilistic polynomial time in the length of its input. For two distribution ensembles $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$ and $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda \in \mathbb{N}}$ indexed by a security parameter $\lambda$, we say that $\mathcal{X}$ and $\mathcal{Y}$ are computationally (respectively, statistically) indistinguishable if for all efficient (respectively, unbounded) adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that for all $\lambda \in \mathbb{N}$:

$$\left|\Pr[\mathcal{A}(1^\lambda, x) = 1 : x \leftarrow \mathcal{X}_\lambda] - \Pr[\mathcal{A}(1^\lambda, y) = 1 : y \leftarrow \mathcal{Y}_\lambda]\right| = \mathsf{negl}(\lambda).$$

**Average min-entropy.** For random variables $X$ and $Y$, the *average min-entropy* of $X$ given $Y$ [DORS08] is defined to be

$$\widetilde{\mathbf{H}}_\infty(X \mid Y) \coloneqq -\log\left(\mathbb{E}_{y \leftarrow Y}\left[\max_x \Pr[X = x \mid Y = y]\right]\right).$$

We will use the following bound on the average min-entropy from [DORS08]:

**Lemma 2.1** (Average Min-Entropy [DORS08, Lemma 2.2])**.** *Let $X, Y$ be random variables and suppose there are at most $2^k$ elements in the support of $Y$. Then, $\widetilde{\mathbf{H}}_\infty(X \mid Y) \geq \mathbf{H}_\infty(X) - k$.*

**Leftover hash lemma.** We recall the (generalized) leftover hash lemma [HILL99, DORS08] that shows how to extract a uniform random value from any source with high average min-entropy. While the general form of the leftover hash lemma can be instantiated with any universal hash function, we describe a specialization that is convenient for our specific application:

**Lemma 2.2** (Generalized Leftover Hash Lemma [DORS08, Lemma 2.4, adapted])**.** *Let $\lambda$ be a security parameter. Let $t, q \in \mathbb{N}$ and $\mathcal{D}$ be any distribution that outputs $\mathbf{x} \in \{0,1\}^t$ and auxiliary information* aux*. Suppose $\widetilde{\mathbf{H}}_\infty(\mathbf{x} \mid \mathsf{aux}) \geq \omega(\log \lambda) + \log q$. Then, there exists a negligible function $\mathsf{negl}(\lambda)$ such that the statistical distance between the following distributions is at most $\mathsf{negl}(\lambda)$:*

$$\left\{(\mathbf{z}^\intercal \mathbf{x}, \mathbf{z}, \mathsf{aux}) : \begin{array}{c}(\mathbf{x}, \mathsf{aux}) \leftarrow \mathcal{D} \\ \mathbf{z} \xleftarrow{R} \mathbb{Z}_q^t\end{array}\right\} \quad \text{and} \quad \left\{(u, \mathbf{z}, \mathsf{aux}) : \begin{array}{c}(\mathbf{x}, \mathsf{aux}) \leftarrow \mathcal{D} \\ \mathbf{z} \xleftarrow{R} \mathbb{Z}_q^t, u \xleftarrow{R} \mathbb{Z}_q\end{array}\right\}$$

**Entropy preservation.** We will also use the fact that if $\mathbf{x} \in \{0,1\}^t$ is a bit-string with high min-entropy, then taking a random subset of the bits of $\mathbf{x}$ still yields a bit-string with high min-entropy of $\mathbf{x}$. We use the statement from [ADW09] (similar statements can also be found in earlier works on randomness extractors [NZ96, Vad06]):

**Lemma 2.3** (Entropy Preservation [ADW09, Lemma A.3, adapted]). *Let $\mathbf{x} \in \{0,1\}^t$ be a random variable. Suppose we sample $I \xleftarrow{R} [t]^k$. Then for all random variables* aux *and all $c > 0$, if $\widetilde{\mathbf{H}}_\infty(\mathbf{x} \mid \mathsf{aux}) \geq \frac{2c}{k} t(1 + \log t) + 3c + 5$, then $\widetilde{\mathbf{H}}_\infty(\mathbf{x}[I] \mid (\mathsf{aux}, I)) \geq c$.*

**ZAPs.** We now recall the formal notion of a statistical ZAP argument [DN00]. We consider two notions of soundness: non-adaptive soundness where the statement is fixed in advance and adaptive soundness where the (malicious) prover can choose the statement *after* it sees the verifier's first message.

**Definition 2.4** (Statistical ZAP Argument). Let $\lambda$ be a security parameter. Let $\mathcal{R}$ be an NP relation defined by a family of polynomial-size circuits $C = \left\{C_n : \{0,1\}^n \times \{0,1\}^h \to \{0,1\}\right\}_{n \in \mathbb{N}}$, where $n$ is the instance length and $h = h(n)$ is a polynomial denoting the witness length. Let $\mathcal{L}$ be the associated NP language. A (public-coin) statistical ZAP argument for $\mathcal{R}$ with public-coin length $t = t(\lambda, n)$ and proof size $\ell = \ell(\lambda, n)$ is a pair of efficient algorithms $(\mathsf{Prove}, \mathsf{Verify})$ with the following syntax:

- $\mathsf{Prove}(1^\lambda, r, x, w) \to \pi$: On input a public random coin $r \in \{0,1\}^t$, a statement $x \in \{0,1\}^n$, and a witness $w \in \{0,1\}^h$, the prove algorithm outputs a proof $\pi \in \{0,1\}^\ell$.
- $\mathsf{Verify}(1^\lambda, r, x, \pi) \to 0/1$: On input a public random coin $r \in \{0,1\}^t$, a statement $x \in \{0,1\}^n$, and a proof $\pi \in \{0,1\}^\ell$, the verification algorithm outputs a bit.

We require $(\mathsf{Prove}, \mathsf{Verify})$ satisfy the following properties:

- **Completeness:** For all $\lambda, n \in \mathbb{N}$, and all $(x, w) \in \mathcal{R}$, it holds that
$$\Pr\left[\mathsf{Verify}(1^\lambda, r, x, \pi) = 1 : r \xleftarrow{R} \{0,1\}^t, \pi \leftarrow \mathsf{Prove}(1^\lambda, r, x, w)\right] = 1.$$
- **Efficiency:** There exists a polynomial $\mathsf{poly}(\cdot,\cdot)$ such that for all $\lambda, n \in \mathbb{N}$ it holds that $t(\lambda, n) \leq \mathsf{poly}(\lambda, n)$ and $\ell(\lambda, n) \leq \mathsf{poly}(\lambda, n)$.
- **Computational non-adaptive soundness:** For all polynomials $n = n(\lambda)$ and all efficient adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that for all $\lambda \in \mathbb{N}$ and all $x \notin \mathcal{L}$, the following holds:
$$\Pr\left[\mathsf{Verify}(1^\lambda, r, x, \pi) = 1 : r \xleftarrow{R} \{0,1\}^t, \pi \leftarrow \mathcal{A}(1^\lambda, r, x)\right] = \mathsf{negl}(\lambda).$$
- **Statistical witness-indistinguishability:** For all polynomials $n = n(\lambda)$, and every (possibly unbounded) adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that $\mathcal{A}$ wins the following game with probability $\mathsf{negl}(\lambda)$:
  1. On input a security parameter $1^\lambda$, the adversary $\mathcal{A}$ chooses $r \in \{0,1\}^t, x \in \{0,1\}^n, w_0, w_1 \in \{0,1\}^h$ to the challenger.
  2. The challenger samples $b \xleftarrow{R} \{0,1\}$, computes $\pi \leftarrow \mathsf{Prove}(1^\lambda, r, x, w_b)$, and sends $\pi$ to the adversary.
  3. The adversary outputs a guess $b' \in \{0,1\}$ and wins the game if all of the following holds:
$$b = b' \quad \text{and} \quad (x, w_0) \in \mathcal{R} \quad \text{and} \quad (x, w_1) \in \mathcal{R}.$$

**Definition 2.5** (Computational Adaptive Soundness). A public-coin statistical ZAP argument for an NP relation $\mathcal{R}$ satisfies computational adaptive soundness if for all polynomials $n = n(\lambda)$, and all efficient adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,
$$\Pr\left[x \notin \mathcal{L} \wedge \mathsf{Verify}(1^\lambda, r, x, \pi) = 1 : r \xleftarrow{R} \{0,1\}^t, (x, \pi) \leftarrow \mathcal{A}(1^\lambda, r)\right] = \mathsf{negl}(\lambda).$$

## 2.1 Lattice Preliminaries

We now recall some preliminaries on lattice-based cryptography. First, we write $D_{\mathbb{Z},s}$ to denote the discrete Gaussian distribution over $\mathbb{Z}$ with width parameter $s > 0$. We will use the following tail bound on discrete Gaussian random variables:

**Lemma 2.6** (Gaussian Tail Bound [MP12, Lemma 2.6, adapted])**.** *For all $\lambda \in \mathbb{N}$,*

$$\Pr[|x| > \sqrt{\lambda}s : x \leftarrow D_{\mathbb{Z},s}] \leq 2^{-\lambda}.$$

**Learning with errors.** Next, we recall the standard learning with errors (LWE) assumption [Reg05]:

**Assumption 2.7** (Learning with Errors [Reg05])**.** Let $\lambda$ be a security parameter and $n = n(\lambda)$, $m = m(\lambda)$, $q = q(\lambda)$, and $s = s(\lambda)$ be lattice parameters. We say that the $\mathsf{LWE}_{n,m,q,s}$ assumption holds if the following two distributions are computationally indistinguishable:

$$\left\{(\mathbf{A}, \mathbf{s}^\intercal\mathbf{A} + \mathbf{e}^\intercal) : \begin{array}{c}\mathbf{A} \xleftarrow{R} \mathbb{Z}_q^{n\times m} \\ \mathbf{s} \xleftarrow{R} \mathbb{Z}_q^n, \mathbf{e} \leftarrow D_{\mathbb{Z},s}^m\end{array}\right\} \quad \text{and} \quad \left\{(\mathbf{A}, \mathbf{u}^\intercal) : \begin{array}{c}\mathbf{A} \xleftarrow{R} \mathbb{Z}_q^{n\times m} \\ \mathbf{u} \xleftarrow{R} \mathbb{Z}_q^m\end{array}\right\}$$

Moreover, for a constant $c > 1$, we say that the $c$-quasi-polynomial $\mathsf{LWE}_{n,m,q,s}$ assumption holds if for all efficient adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the advantage in distinguishing the distributions is bounded by $2^{-\log^c(\lambda)} \cdot \mathsf{negl}(\lambda)$.

**Shifted multi-preimage trapdoor sampler.** Next, we recall the notion of a shifted multi-preimage trapdoor sampler introduced in [WWW25]:

**Definition 2.8** (Shifted Multi-Preimage Trapdoor Sampler [WWW25, adapted])**.** Let $\lambda$ be a security parameter and $\ell$ be a dimension. Let $n = n(\lambda, \ell)$, $t = t(\lambda, \ell)$, $q = q(\lambda, \ell)$, and $s = s(\lambda, \ell)$ be functions. An $(n, t, q, s)$-shifted multi-preimage trapdoor sampler is a pair of efficient algorithms $(\mathsf{Gen}, \mathsf{Expand}, \mathsf{SampleMultPre})$ with the following syntax:

- $\mathsf{Gen}(1^\lambda, 1^\ell) \to \mathsf{crs}$: On input the security parameter $\lambda$ and the dimension $\ell$, the generator algorithm outputs a common reference string crs.
- $\mathsf{Expand}(1^\lambda, 1^\ell, \mathsf{crs}) \to (\mathbf{A}_1, \ldots, \mathbf{A}_\ell, \mathsf{td})$: On input the security parameter $\lambda$, the dimension $\ell$, and the common reference string crs, the expand algorithm outputs a collection of matrices $\mathbf{A}_i \in \mathbb{Z}_q^{n\times t}$ and a trapdoor td. This algorithm is *deterministic*.
- $\mathsf{SampleMultPre}(\mathsf{td}, \mathbf{t}_1, \ldots, \mathbf{t}_\ell) \to (\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_\ell, \mathbf{c})$: On input a trapdoor td and a collection of preimages $\mathbf{t}_1, \ldots, \mathbf{t}_\ell$, the shifted multi-preimage sampling algorithm outputs a shift $\mathbf{c} \in \mathbb{Z}_q^n$ together with preimages $\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_\ell \in \mathbb{Z}_q^t$.

The structured trapdoor sampler should satisfy the following properties:

- **Correctness:** For all $\lambda, \ell \in \mathbb{N}$, all crs in the support of $\mathsf{Gen}(1^\lambda, 1^\ell)$, all target vectors $\mathbf{t}_1, \ldots, \mathbf{t}_\ell \in \mathbb{Z}_q^n$, and setting $(\mathbf{A}_1, \ldots, \mathbf{A}_\ell, \mathsf{td}) = \mathsf{Expand}(1^\lambda, 1^\ell, \mathsf{crs})$, it holds that:

  $$\Pr[\forall i \in [\ell]\ \mathbf{A}_i\boldsymbol{\pi}_i = \mathbf{t}_i + \mathbf{c}] = 1,$$

  where the probability is over $(\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_\ell, \mathbf{c}) \leftarrow \mathsf{SampleMultPre}(\mathsf{td}, \mathbf{t}_1, \ldots, \mathbf{t}_\ell)$.
- **Preimage norm bound:** For all polynomials $\ell = \ell(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all crs in the support of $\mathsf{Gen}(1^\lambda, 1^\ell)$, setting $(\mathbf{A}_1, \ldots, \mathbf{A}_\ell, \mathsf{td}) = \mathsf{Expand}(1^\lambda, 1^\ell, \mathsf{crs})$, and for all target vectors $\mathbf{t}_1, \ldots, \mathbf{t}_\ell \in \mathbb{Z}_q^n$:

  $$\Pr\left[\|\boldsymbol{\pi}_i\| \geq \sqrt{\ell t}s : (\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_\ell, \mathbf{c}) \leftarrow \mathsf{SampleMultPre}(\mathsf{td}, \mathbf{t}_1, \ldots, \mathbf{t}_\ell)\right] = \mathsf{negl}(\lambda).$$

- **Somewhere programmable:** There exists an efficient algorithm GenProg such that for all polynomials $\ell = \ell(\lambda)$, the following holds:
  - For all $\lambda \in \mathbb{N}$, all indices $i \in [\ell]$, and all matrices $\mathbf{A}_i \in \mathbb{Z}_q^{n \times t}$, it holds that
$$\Pr\left[\mathbf{A}_i = \widetilde{\mathbf{A}}_i : \begin{array}{c} \widetilde{\mathsf{crs}} \leftarrow \mathsf{GenProg}(1^\lambda, 1^\ell, i, \mathbf{A}_i) \\ (\widetilde{\mathbf{A}}_1, \ldots, \widetilde{\mathbf{A}}_\ell, \widetilde{\mathbf{T}}) = \mathsf{Expand}(1^\lambda, 1^\ell, \widetilde{\mathsf{crs}}) \end{array}\right] = 1.$$
  - There exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$ and all $i \in [\ell]$, the statistical distance between the following distributions is $\mathsf{negl}(\lambda)$:
$$\left\{\mathsf{crs} : \mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda, 1^\ell)\right\} \quad \text{and} \quad \left\{\widetilde{\mathsf{crs}} : \begin{array}{c} \mathbf{A}_i \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times t} \\ \widetilde{\mathsf{crs}} \leftarrow \mathsf{GenProg}(1^\lambda, 1^\ell, i, \mathbf{A}_i) \end{array}\right\}.$$
    When these distributions are identical, we say the shifted multi-preimage trapdoor sampler satisfies *perfect* somewhere programmability.
- **Transparent setup:** The common reference string crs output by Gen is exactly the random coins given to Gen.

**Remark 2.9** (Preimage Distribution Property)**.** The shifted multi-preimage trapdoor sampler definition from [WWW25] states a different preimage property which they call *preimage distribution*. Their property asserts that the outputs $(\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_\ell, \mathbf{c}) \leftarrow \mathsf{SampleMultPre}(\mathsf{td}, \mathbf{t}_1, \ldots, \mathbf{t}_\ell)$ are statistically close to the distribution obtained by sampling $\mathbf{c} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$ and $\boldsymbol{\pi}_i \leftarrow (\mathbf{A}_i)_s^{-1}(\mathbf{t}_i + \mathbf{c})$. However, this property only holds with overwhelming probability over the choice of $\mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda, 1^\ell)$. In contrast, the property we state here holds for *all* common reference strings, which is crucial for our application since the CRS may be chosen adversarially. Finally, it is easy to see that the construction of [WWW25], satisfies the preimage bound property. Specifically, for *every* choice of CRS (which determines the matrices $\mathbf{A}_1, \ldots, \mathbf{A}_\ell$), the work of [WWW25, Construction 4.6] shows how to construct a trapdoor $\mathbf{T}$ where $\|\mathbf{T}\| = 1$ and
$$\underbrace{\left[\begin{array}{ccc|c} \mathbf{A}_1 & & & \mathbf{G} \\ & \ddots & & \vdots \\ & & \mathbf{A}_\ell & \mathbf{G} \end{array}\right]}_{\mathbf{D}} \cdot \mathbf{T} = \mathbf{I}_\ell \otimes \mathbf{G},$$
where $\mathbf{G} \in \mathbb{Z}_q^{n \times n\lceil \log q \rceil}$ is the gadget matrix [MP12]. The SampleMultPre algorithm uses $\mathbf{T}$ to sample a Gaussian-distributed preimage $\mathbf{D}_s^{-1}(\mathbf{t})$ where where $\mathbf{t} \in \mathbb{Z}_q^{n\ell}$ is the vertical concatenation of $\mathbf{t}_1, \ldots, \mathbf{t}_\ell \in \mathbb{Z}_q^n$. Since the matrix $\mathbf{D}$ has a "gadget trapdoor" $\mathbf{T}$ with norm 1 (irrespective of the choice of the CRS), we can appeal to a standard Gaussian tail bound (c.f., [MP12, Lemma 2.6]) to conclude that the norm of the preimage is bounded with overwhelming probability (for the parameter choices from [WWW25, Theorem 4.7]).

**Theorem 2.10** (Shifted Multi-Preimage Trapdoor Sampler [WWW25, Theorem 4.7])**.** *Let $n = n(\lambda, \ell)$, $q = q(\lambda, \ell)$ be arbitrary non-negative functions where $n \geq \lambda$. Let $m = 3n\lceil \log q \rceil$ and $t = m(\lceil \log q \rceil + 1)$. Then for all $s \geq (\ell t + m)\log(\ell n)$, there exists an $(n, q, t, s)$-shifted multi-preimage trapdoor sampler* $(\mathsf{Gen}, \mathsf{Expand}, \mathsf{SampleMultPre})$ *with perfect somewhere programmability and transparent setup. The size of the CRS output by* $\mathsf{Gen}(1^\lambda, 1^\ell)$ *is* $nt \log q$.

# 3 Interactive Hidden-Bits Generator

The work of [CKSU21] introduce the notion of an *interactive hidden bits generator* to construct (statistical) ZAPs from pairings groups. Interactive hidden-bits are a generalization of standard (non-interactive) dual-mode hidden-bits generators [QRW19, LPWW20].

At a high level, an interactive hidden-bits generator is a two-message protocol that allows a prover and a verifier to jointly sample a hidden-bits string. The verifier is public-coin and sends the first message, which is simply a uniform random string (i.e., its random coins). The prover responds with a message. At the conclusion of the protocol, the prover receives the hidden-bits string while the verifier receives a commitment to the hidden-bits string. The prover can then open a subset of the bits to the verifier. The main requirements are as follows:

- **Hiding:** First, the unopened bits should remain hidden (and statistically close to uniform) from the perspective of the verifier.
- **Binding:** Second, the prover should not be able to significantly bias the distribution of the hidden-bits string. To formally capture this, the work of [CKSU21] introduces a trapdoor algorithm $\mathsf{TrapCoin}$ that outputs a string $r^*$ (which is computationally indistinguishable from the verifier's first message) together with a trapdoor $\mathsf{td}$. The trapdoor $\mathsf{td}$ can be used to extract the hidden-bits string from the commitment. Of course, if $\mathsf{TrapCoin}$ always succeeds, or even succeeds with non-negligible probability, then it violates the hiding property. For this reason, we require it to succeed with a negligible, but not too small, probability $\mu$ and refer to this as a $\mu$-extraction property.
- **Succinctness:** Finally, the succinctness property asserts that the size of the commitment should be much shorter than the length of the hidden-bits string.

**Definition 3.1** (Interactive Hidden-Bits Generator [CKSU21])**.** Let $\lambda$ be a security parameter and $\ell$ be an output length parameter, i.e., the number of hidden bits. An interactive hidden-bits generator with public-coin length $\ell_{\mathsf{pub}} = \ell_{\mathsf{pub}}(\lambda, \ell)$ and commitment length $\ell_{\mathsf{com}} = \ell_{\mathsf{com}}(\lambda, \ell)$ is a tuple of efficient algorithms $\Pi_{\mathsf{IHBG}} = (\mathsf{GenBits}, \mathsf{Verify})$ with the following syntax:

- $\mathsf{GenBits}(1^\lambda, 1^\ell, r) \to (\mathsf{com}, \pi_1, \ldots, \pi_\ell, \boldsymbol{\rho})$: On input a security parameter $\lambda$, a length parameter $\ell$, and a public-coin $r \in \{0,1\}^{\ell_{\mathsf{pub}}}$, the bit-generation algorithm outputs a commitment $\mathsf{com} \in \{0,1\}^{\ell_{\mathsf{com}}}$, openings $\pi_1, \ldots, \pi_\ell$, and a hidden-bits string $\boldsymbol{\rho} \in \{0,1\}^\ell$.
- $\mathsf{Verify}(r, \mathsf{com}, i, b, \pi_i) \to \beta$: On input a public-coin $r \in \{0,1\}^{\ell_{\mathsf{pub}}}$, a commitment $\mathsf{com} \in \{0,1\}^{\ell_{\mathsf{com}}}$, an index $i \in [\ell]$, a bit $b \in \{0,1\}$ and an opening $\pi$, the verification algorithm outputs a bit $\beta \in \{0,1\}$.

We require $\Pi_{\mathsf{IHBG}}$ to satisfy the following properties:

- **Correctness:** For all $\lambda \in \mathbb{N}$, $\ell \in \mathbb{N}$, $i \in [\ell]$, and $r \in \{0,1\}^{\ell_{\mathsf{pub}}}$, the following holds:

$$\Pr\left[\mathsf{Verify}(r, \mathsf{com}, i, \rho_i, \pi_i) = 1 : (\mathsf{com}, \pi_1, \ldots, \pi_\ell, \boldsymbol{\rho}) \leftarrow \mathsf{GenBits}(1^\lambda, 1^\ell, r)\right] = 1.$$

- **Succinctness:** There exists a fixed polynomial $\mathsf{poly}(\cdot,\cdot)$ such that commitment length $\ell_{\mathsf{com}}(\lambda, \ell) \le \mathsf{poly}(\lambda, \log \ell)$.
- **Statistical hiding:** For a security parameter $\lambda$, a length parameter $\ell$, a bit $b \in \{0,1\}$, a simulator $\mathsf{Sim}$, and an adversary $\mathcal{A}$, we define the hiding game as follows:

  1. On input a security parameter $1^\lambda$ and length parameter $1^\ell$, the adversary $\mathcal{A}$ chooses a public coin $r \in \{0,1\}^{\ell_{\mathsf{pub}}}$, and sends $r$ to the challenger.
  2. If $b = 0$, the challenger samples $(\mathsf{com}, \pi_1, \ldots, \pi_\ell, \boldsymbol{\rho}) \leftarrow \mathsf{GenBits}(1^\lambda, 1^\ell, r)$. If $b = 1$, the challenger samples $\boldsymbol{\rho} \xleftarrow{\mathrm{R}} \{0,1\}^\ell$. The challenger sends $\boldsymbol{\rho}$ to $\mathcal{A}$.
  3. The adversary $\mathcal{A}$ chooses a subset $S \subseteq [\ell]$ and sends it to the challenger.
  4. If $b = 1$, then the challenger computes $(\mathsf{com}, (\pi_i)_{i \in S}) \leftarrow \mathsf{Sim}(1^\lambda, 1^\ell, r, S, \boldsymbol{\rho}_S)$, where $\boldsymbol{\rho}_S$ denotes the bits of $\boldsymbol{\rho}$ associated with the indices $i \in S$. In both cases, the challenger responds with $(\mathsf{com}, (\pi_i)_{i \in S})$.
  5. The adversary $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$.

  We say that $\Pi_{\mathsf{IHBG}}$ is statistically hiding if there exists a (possibly unbounded) simulator $\mathsf{Sim}$, such that for all polynomials $\ell = \ell(\lambda)$ and all (possibly unbounded) adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$\left|\Pr\left[b' = 1 : b = 0\right] - \Pr\left[b' = 1 : b = 1\right]\right| \le \mathsf{negl}(\lambda).$$

- **Extraction:** We additionally require the following efficient algorithms:
  - $\mathsf{TrapCoin}(1^\lambda, 1^\ell) \to (r^*, \mathsf{td})$: On input a security parameter $\lambda$ and a length parameter $\ell$, the trapdoor public-coin generation algorithm outputs a public-coin $r^* \in \{0,1\}^{\ell_{\mathsf{pub}}}$ and a trapdoor $\mathsf{td}$.

– $\mathsf{Extract}(\mathsf{td}, \mathsf{com}) \to \boldsymbol{\rho}$: On input a trapdoor $\mathsf{td}$ and a commitment $\mathsf{com} \in \{0,1\}^{\ell_{\mathsf{com}}}$, the extraction algorithm outputs a string $\boldsymbol{\rho} \in \{0,1\}^{\ell}$.

We require the following properties:

– **Mode indistinguishability:** For all efficient adversaries $\mathcal{A}$ and all polynomials $\ell = \ell(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that

$$\left|\Pr\left[\mathcal{A}(1^\lambda, 1^\ell, r) = 1 : r \xleftarrow{R} \{0,1\}^{\ell_{\mathsf{pub}}}\right] - \Pr\left[\mathcal{A}(1^\lambda, 1^\ell, r^*) = 1 : (r^*, \mathsf{td}) \leftarrow \mathsf{TrapCoin}(1^\lambda, 1^\ell)\right]\right| = \mathsf{negl}(\lambda).$$

Moreover, for a constant $c > 1$, we say that $\Pi_{\mathsf{lHBG}}$ satisfies $c$-quasi-polynomial mode indistinguishability if for all efficient adversaries $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the above distinguishing advantage is bounded by $2^{-\log^c(\lambda)} \cdot \mathsf{negl}(\lambda)$.

– $\mu$-**Extraction:** For an adversary $\mathcal{A}$, a security parameter $\lambda$, and a length parameter $\ell$, we define the extraction game as follows:

* The challenger samples $(r^*, \mathsf{td}) \leftarrow \mathsf{TrapCoin}(1^\lambda, 1^\ell)$ sends $r^*$ to the adversary.
* The adversary $\mathcal{A}$ either aborts (with output $\perp$) or outputs a commitment $\mathsf{com}$, a set of indices $S \subseteq [\ell]$, and a tuple of openings $(\pi_i)_{i \in S}$.

Define the following events:

* Let $E_{\mathsf{lose}}$ be the event that for all $i \in S$ it holds that $\mathsf{Verify}(r^*, \mathsf{com}, i, 1 - \rho_i, \pi_i) = 0$, where $\boldsymbol{\rho} \leftarrow \mathsf{Extract}(\mathsf{td}, \mathsf{com})$.
* Let $E_\top$ be the event that $\mathcal{A}$ does not abort.

Let $\mu(\lambda)$ be any positive function of $\lambda$. We say that $\Pi_{\mathsf{lHBG}}$ is $\mu$-extractable if for all polynomials $\ell = \ell(\lambda)$ and all efficient algorithms $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$\Pr\left[E_\top \ \wedge \ E_{\mathsf{lose}}\right] \geq \mu(\lambda) \cdot \Big(\Pr\left[E_\top\right] - \mathsf{negl}(\lambda)\Big).$$

**Theorem 3.2** (Statistical ZAP Arguments [CKSU21])**.** *Suppose there exists an interactive hidden-bits generator that satisfies correctness, succinctness, statistical hiding and $\mu$-extraction for some negligible function $\mu(\lambda) = \mathsf{negl}(\lambda)$. Then there exists a (public-coin) statistical ZAP argument for* NP *with non-adaptive soundness.*

**Remark 3.3** (Adaptive Security)**.** If the interactive hidden-bits generator is sub-exponentially secure, then the ZAP from Theorem 3.2 is also sub-exponentially (non-adaptively) sound. Using standard complexity leveraging, we can get an adaptively-sound ZAP for bounded-length statements from a sub-exponentially sound ZAP in a black-box way. Looking ahead, this means that our result can be extended to getting a statistical ZAP argument for NP with adaptive soundness assuming sub-exponential hardness of LWE (with a polynomial modulus-to-noise ratio).

**Single-bit hiding.** Similar to [Wat24], we define a single-bit hiding property for an interactive hidden-bits generator and then show that it implies the hiding property from Definition 3.1. This is a simpler property to analyze.

**Definition 3.4** (Single-Bit Hiding)**.** Let $\Pi_{\mathsf{lHBG}} = (\mathsf{GenBits}, \mathsf{Verify})$ be an interactive hidden-bits generator. For an adversary $\mathcal{A}$, a security parameter $\lambda$, a length parameter $\ell$ and a bit $b \in \{0,1\}$, we define the single bit hiding game as follows:

1. On input a security parameter $1^\lambda$ and a length parameter $1^\ell$, the adversary $\mathcal{A}$ chooses public-coin $r \in \{0,1\}^{\ell_{\mathsf{pub}}}$ and an index $i^* \in [\ell]$, and sends $(r, i^*)$ to the challenger.
2. The challenger samples $(\mathsf{com}, \pi_1, \ldots, \pi_\ell, \boldsymbol{\rho}) \leftarrow \mathsf{GenBits}(1^\lambda, 1^\ell, r)$.
3. If $b = 0$, the challenger sets $\beta = \rho_{i^*}$. If $b = 1$, the challenger samples a bit $\beta \xleftarrow{R} \{0,1\}$.
4. The challenger sends $\big(\mathsf{com}, (\rho_i, \pi_i)_{i \neq i^*}, \beta\big)$ to $\mathcal{A}$.

5. The adversary $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$.

We say that $\Pi_{\mathsf{lHBG}}$ satisfies single-bit hiding if for all polynomials $\ell = \ell(\lambda)$ and all (possibly unbounded) adversaries $\mathcal{A}$, there exists a negligible function such that

$$|\Pr[b' = 0 : b = 0] - \Pr[b' = 0 : b = 1]| \leq \mathsf{negl}(\lambda)$$

in the single-bit hiding game.

**Theorem 3.5** (Single-Bit Hiding to Statistical Hiding)**.** *If $\Pi_{\mathsf{lHBG}}$ satisfies single bit hiding, it satisfies statistical hiding.*

*Proof.* We define an (unbounded) simulator algorithm $\mathsf{Sim}$ for the statistical hiding game:.

1. On input $(1^\lambda, 1^\ell, r, S, \boldsymbol{\rho}_S)$, the simulator starts by sampling $(\widetilde{\mathsf{com}}, \tilde{\pi}_1, \ldots, \tilde{\pi}_\ell, \tilde{\boldsymbol{\rho}}) \leftarrow \mathsf{GenBits}(1^\lambda, 1^\ell, r)$ conditioned on $\tilde{\rho}_i = \rho_i$ for all $i \in S$. If there does not exist any tuple $(\widetilde{\mathsf{com}}, \tilde{\pi}_1, \ldots, \tilde{\pi}_\ell, \tilde{\boldsymbol{\rho}})$ that satisfies this condition in the support of $\mathsf{GenBits}(1^\lambda, 1^\ell, r)$, then the simulator aborts with output $\perp$.
2. The simulator outputs $(\widetilde{\mathsf{com}}, (\tilde{\pi}_i)_{i \in S})$.

Let $\mathcal{A}$ be an adversary for $\Pi_{\mathsf{lHBG}}$ in the statistical hiding game. For each $k \in [\ell + 1]$, we define a hybrid experiment $\mathsf{Hyb}_k$ between the challenger and the adversary $\mathcal{A}$ as follows:

1. On input the security parameter $1^\lambda$ and the length parameter $1^\ell$, the adversary $\mathcal{A}$ chooses a public-coin $r \in \{0,1\}^{\ell_{\mathsf{pub}}}$, and sends $r$ to the challenger.
2. The challenger samples $\rho_i \xleftarrow{\mathrm{R}} \{0,1\}$ for $i \in [k-1]$ and $(\mathsf{com}, \pi_1, \ldots, \pi_\ell, \boldsymbol{\rho}') \leftarrow \mathsf{GenBits}(1^\lambda, 1^\ell, r)$. Then, the challenger gives the string $\rho_1 \cdots \rho_{k-1} \rho'_k \cdots \rho'_\ell$ to $\mathcal{A}$.
3. The adversary $\mathcal{A}$ chooses a subset $S \subseteq [\ell]$ and sends it to the challenger.
4. The challenger samples $(\widetilde{\mathsf{com}}, \tilde{\pi}_1, \ldots, \tilde{\pi}_\ell, \tilde{\boldsymbol{\rho}}) \leftarrow \mathsf{GenBits}(1^\lambda, 1^\ell, r)$ conditioned on

$$\tilde{\rho}_i = \begin{cases} \rho_i & i < k \\ \rho'_i & i \geq k \end{cases}$$

   for all $i \in S$. If there does not exist any tuple $(\widetilde{\mathsf{com}}, \tilde{\pi}_1, \ldots, \tilde{\pi}_\ell, \tilde{\boldsymbol{\rho}})$ in the support of $\mathsf{GenBits}(1^\lambda, 1^\ell, r)$ that satisfies this condition, then the challenger responds to $\mathcal{A}$ with $\perp$.
5. The challenger responds to $\mathcal{A}$ with $(\widetilde{\mathsf{com}}, (\tilde{\pi}_i)_{i \in S})$.
6. The adversary $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$.

We write $\mathsf{Hyb}_i(\mathcal{A})$ to denote the random variable corresponding to the output $\mathcal{A}$ in $\mathsf{Hyb}_i$. By construction, we have the following:

- In $\mathsf{Hyb}_1$, the challenger samples $(\mathsf{com}, \pi_1, \ldots, \pi_\ell, \boldsymbol{\rho}') \leftarrow \mathsf{GenBits}(1^\lambda, 1^\ell, r)$, so $\boldsymbol{\rho}'$ is generated correctly according to $\mathsf{GenBits}(1^\lambda, 1^\ell, r)$. In this experiment, the challenger gives the hidden-bits string $\boldsymbol{\rho}'$ to $\mathcal{A}$. After the adversary chooses a subset $S \subseteq [\ell]$, the challenger then samples $(\widetilde{\mathsf{com}}, \tilde{\pi}_1, \ldots, \tilde{\pi}_\ell, \tilde{\boldsymbol{\rho}}) \leftarrow \mathsf{GenBits}(1^\lambda, 1^\ell, r)$ conditioned on $\tilde{\boldsymbol{\rho}}_S = \boldsymbol{\rho}'_S$. Since the marginal distribution of $\boldsymbol{\rho}'_S$ is distributed exactly as $(\mathsf{com}, \pi_1, \ldots, \pi_\ell, \boldsymbol{\rho}') \leftarrow \mathsf{GenBits}(1^\lambda, 1^\ell, r)$, the joint distribution of $\widetilde{\mathsf{com}}$ and $(\pi_i)_{i \in S}$ is distributed exactly as a fresh sample from $\mathsf{GenBits}(1^\lambda, 1^\ell, r)$. Thus, this game is equivalent to the statistical hiding game where $b = 0$.
- In $\mathsf{Hyb}_{\ell+1}$, the challenger samples $\boldsymbol{\rho} \xleftarrow{\mathrm{R}} \{0,1\}^\ell$ and gives $\boldsymbol{\rho}$ to $\mathcal{A}$ as the hidden-bit string. After the adversary $\mathcal{A}$ outputs a set $S \subseteq [\ell]$, the challenger samples $(\widetilde{\mathsf{com}}, \tilde{\pi}_1, \ldots, \tilde{\pi}_\ell, \tilde{\boldsymbol{\rho}}) \leftarrow \mathsf{GenBits}(1^\lambda, 1^\ell, r)$ conditioned on $\tilde{\boldsymbol{\rho}}_S = \boldsymbol{\rho}_S$ where $\boldsymbol{\rho} \xleftarrow{\mathrm{R}} \{0,1\}^\ell$. This exactly coincides with the behavior of the simulation algorithm $\mathsf{Sim}$. Since the challenger samples $\boldsymbol{\rho} \xleftarrow{\mathrm{R}} \{0,1\}^\ell$ in this experiment, the output distribution is equivalent to the statistical hiding game where $b = 1$.

We now use $\mathcal{A}$ to construct an (unbounded) adversary $\mathcal{B}$ for the single-bit hiding game:

1. On input a security parameter $1^\lambda$ and length parameter $1^\ell$, algorithm $\mathcal{B}$ runs $\mathcal{A}$ on $(1^\lambda, 1^\ell)$. Algorithm $\mathcal{A}$ outputs a public-coin $r \in \{0,1\}^{\ell_{\mathsf{pub}}}$. Algorithm $\mathcal{B}$ samples an index $k^* \xleftarrow{R} [\ell]$ and forwards the public-coin $r$ and the index $k^*$ to the challenger.
2. The challenger replies to $\mathcal{B}$ with $\big(\mathsf{com}, (\rho'_i, \pi'_i)_{i \neq k^*}, \beta\big)$.
3. Algorithm $\mathcal{B}$ samples $\rho_i \xleftarrow{R} \{0,1\}$ for $i \in [k^* - 1]$ and gives the string $\rho_1, \ldots, \rho_{k^*-1}, \beta, \rho'_{k^*+1}, \ldots, \rho'_\ell$ to $\mathcal{A}$.
4. The adversary $\mathcal{A}$ chooses a subset $S \subseteq [\ell]$ and sends it to algorithm $\mathcal{B}$.
5. Algorithm $\mathcal{B}$ samples $(\widetilde{\mathsf{com}}, \tilde{\pi}_1, \ldots, \tilde{\pi}_\ell, \tilde{\boldsymbol{\rho}}) \leftarrow \mathsf{GenBits}(1^\lambda, 1^\ell, r)$ conditioned on
$$\tilde{\rho}_i = \begin{cases} \rho_i & i < k^* \\ \beta & i = k^* \\ \rho'_i & i > k^* \end{cases}$$
for all $i \in S$. If there does not exist any tuple $(\widetilde{\mathsf{com}}, \tilde{\pi}_1, \ldots, \tilde{\pi}_\ell, \tilde{\boldsymbol{\rho}})$ in the support of $\mathsf{GenBits}(1^\lambda, 1^\ell, r)$ that satisfies this condition, then algorithm $\mathcal{B}$ responds to $\mathcal{A}$ with $\perp$.
6. Algorithm $\mathcal{B}$ responds to $\mathcal{A}$ with $(\widetilde{\mathsf{com}}, (\tilde{\pi}_i)_{i \in S})$.
7. The adversary $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$, which $\mathcal{B}$ outputs to the challenger.

In the single-bit hiding experiment, the challenger samples $(\mathsf{com}, \pi_1, \ldots, \pi_\ell, \boldsymbol{\rho}') \leftarrow \mathsf{GenBits}(1^\lambda, 1^\ell, r)$. Take any fixed index $k \in [\ell]$. Then conditioned on $k^* = k$, the following holds:

- If $b = 0$, the challenger sets $\beta = \rho'_k$. Then, when $k^* = k$, algorithm $\mathcal{B}$ perfectly simulates an execution of $\mathsf{Hyb}_k$ for algorithm $\mathcal{A}$, and hence outputs 1 with probability $\Pr[\mathsf{Hyb}_k(\mathcal{A}) = 1]$.
- If $b = 1$, the challenger samples $\beta \xleftarrow{R} \{0,1\}$. Then, when $k^* = k$, algorithm $\mathcal{B}$ perfectly simulates an execution of $\mathsf{Hyb}_{k+1}$ for algorithm $\mathcal{A}$, and hence outputs 1 with probability $\Pr[\mathsf{Hyb}_{k+1}(\mathcal{A}) = 1]$.

Since $k^*$ is uniform over $[\ell]$, then for any $k \in [\ell]$, we conclude that $\Pr[k^* = k] = 1/\ell$. The distinguishing advantage of $\mathcal{B}$ is then

$$\left| \frac{1}{\ell} \sum_{k \in [\ell]} \left( \Pr[\mathsf{Hyb}_{k+1}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_k(\mathcal{A}) = 1] \right) \right| = \frac{1}{\ell} \cdot \left| \Pr[\mathsf{Hyb}_{\ell+1}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1] \right|.$$

By single-bit hiding of $\Pi_{\mathsf{IHBG}}$, $\frac{1}{\ell} \cdot \left| \Pr[\mathsf{Hyb}_{\ell+1}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1] \right| \leq \mathsf{negl}(\lambda)$ for some negligible function $\mathsf{negl}(\cdot)$. Correspondingly,

$$\left| \Pr[\mathsf{Hyb}_{\ell+1}(\mathcal{A})] - \Pr[\mathsf{Hyb}_1(\mathcal{A})] \right| \leq \ell \cdot \mathsf{negl}(\lambda),$$

which is negligible since $\ell = \mathsf{poly}(\lambda)$. As argued previously, $\mathsf{Hyb}_1$ corresponds to the statistical hiding experiment with bit $b = 0$ while $\mathsf{Hyb}_{\ell+1}$ corresponds to the statistical hiding experiment with bit $b = 1$. The theorem follows. □

# 4 Constructing an Interactive Hidden-Bits Generator from Lattices

We now show how to use a shifted multi-preimage trapdoor sampler (Definition 2.8) to construct an interactive hidden-bits generator (Definition 3.1).

**Construction 4.1.** Let $\lambda$ be a security parameter and $\ell$ be a length parameter. We start by defining the following parameters:

- Let $\Pi_{\mathsf{samp}} = (\mathsf{Gen}, \mathsf{Expand}, \mathsf{SampleMultPre})$ be a $(n, t, q, s)$-shifted multi-preimage trapdoor sampler, where $n, t, q, s$ are functions of $\lambda$ and $\ell$. We assume $\Pi_{\mathsf{samp}}$ has a transparent setup procedure, and moreover, that the length of the (public) randomness needed by Gen is $\ell_{\mathsf{samp}} = \ell_{\mathsf{samp}}(\lambda, \ell)$.
- Let $s_{\mathsf{LWE}} = s_{\mathsf{LWE}}(\lambda, \ell)$ be a Gaussian width parameter.
- Let $N = N(\lambda, q)$ be the number of indices used for prover re-randomization.
- Let $B_{\mathsf{max}} = B_{\mathsf{max}}(\lambda, \ell)$ be a norm bound and $B_{\mathsf{round}} = B_{\mathsf{round}}(\lambda, \ell)$ be a rounding boundary parameter.
- Let $\ell_{\mathsf{pub}} = \ell_{\mathsf{samp}} + \ell t(\lceil \log q \rceil + \lambda)$ be the public-coin length. For each $r \in \{0,1\}^{\ell_{\mathsf{pub}}}$, we derive an associated tuple $(\mathsf{crs}_{\mathsf{samp}}, \tilde{\mathbf{v}}_1, \ldots, \tilde{\mathbf{v}}_\ell)$ from $r$ as follows:
  1. First, parse $r = r_{\mathsf{samp}} \| r_1 \| \cdots \| r_\ell$ where $r_{\mathsf{samp}} \in \{0,1\}^{\ell_{\mathsf{samp}}}$ and $r_i \in \{0,1\}^{t(\lceil \log q \rceil + \lambda)}$ for all $i \in [\ell]$.
  2. Let $\mathsf{crs}_{\mathsf{samp}} = \mathsf{Gen}(1^\lambda, 1^\ell; r_{\mathsf{samp}})$.
  3. For each $i \in [\ell]$, we further parse $r_i = r_{i,1} \| \cdots \| r_{i,t}$ where $r_{i,j} \in \{0,1\}^{\lceil \log q \rceil + \lambda}$. We now interpret $r_{i,j}$ as the binary representation of a $(\lceil \log q \rceil + \lambda)$-bit integer and define the vector $\tilde{\mathbf{v}}_i \in \mathbb{Z}_q^t$ where $\tilde{v}_{i,j} = r_{i,j} \bmod q$ for all $j \in [t]$.

We now construct a interactive hidden-bits generator $\Pi_{\mathsf{IHBG}} = (\mathsf{GenBits}, \mathsf{Verify})$ with public-coin length $\ell_{\mathsf{pub}}$ as follows:

- $\mathsf{GenBits}(1^\lambda, 1^\ell, r)$: On input a security parameter $\lambda$, a length parameter $\ell$, and a public coin $r \in \{0,1\}^{\ell_{\mathsf{pub}}}$, the bit-generation algorithm proceeds as follows:
  1. Let $(\mathsf{crs}_{\mathsf{samp}}, \tilde{\mathbf{v}}_1, \ldots, \tilde{\mathbf{v}}_\ell)$ be the tuple associated with the public coin $r$. Expand $\mathsf{crs}_{\mathsf{samp}}$ by computing $(\mathbf{A}_1, \ldots, \mathbf{A}_\ell, \mathsf{td}_{\mathsf{samp}}) = \mathsf{Expand}(1^\lambda, 1^\ell, \mathsf{crs}_{\mathsf{samp}})$.
  2. Sample indices $i_1, \ldots, i_N \xleftarrow{\mathrm{R}} [t]$ and $z_1, \ldots, z_N \xleftarrow{\mathrm{R}} \mathbb{Z}_q$. Compute $\mathbf{v}_i = \tilde{\mathbf{v}}_i + \sum_{j \in [N]} z_j \boldsymbol{\eta}_{i_j}$ for $i \in [\ell]$, where $\boldsymbol{\eta}_i \in \mathbb{Z}_q^t$ denotes the $i^{\text{th}}$ standard basis vector.
  3. Repeat the following procedure up to $\lambda$ times:
     (a) Sample $\mathbf{w}_1, \ldots, \mathbf{w}_\ell \xleftarrow{\mathrm{R}} \{0,1\}^t$.
     (b) Compute $(\boldsymbol{\pi}'_1, \ldots, \boldsymbol{\pi}'_\ell, \mathbf{c}) \leftarrow \mathsf{SampleMultPre}(\mathsf{td}_{\mathsf{samp}}, \mathbf{A}_1 \mathbf{w}_1, \ldots, \mathbf{A}_\ell \mathbf{w}_\ell)$.
     (c) For each $i \in [\ell]$, set $\boldsymbol{\pi}_i = \boldsymbol{\pi}'_i - \mathbf{w}_i$. If $\|\boldsymbol{\pi}_i\| > B_{\mathsf{max}}$ then set $\rho_i = \perp$. Otherwise, compute $u_i = \mathbf{v}_i^{\mathsf{T}} \boldsymbol{\pi}_i$ and set $\rho_i$ as follows:
$$\rho_i = \begin{cases} 0 & u_i \in [-B_{\mathsf{round}}, B_{\mathsf{round}}] \\ 1 & u_i \in \left[ \lfloor q/2 \rfloor - B_{\mathsf{round}}, \lfloor q/2 \rfloor + B_{\mathsf{round}} \right] \\ \perp & \text{otherwise} \end{cases}$$
     (d) If $\rho_i = \perp$ for some $i \in [\ell]$, then repeat the procedure. Otherwise, declare the procedure successful and continue.
  4. If the procedure does not succeed after $\lambda$ iterations, it sets $\mathbf{c} = \perp$ and $\pi_i = \perp$ and $\rho_i = 0$ for all $i \in [\ell]$. Finally, the algorithm sets $\boldsymbol{\rho} = \rho_1 \cdots \rho_\ell \in \{0,1\}^\ell$ and outputs $\mathsf{com} = (i_1, \ldots, i_N, z_1, \ldots, z_N, \mathbf{c})$, the openings $\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_\ell$, and the string $\boldsymbol{\rho}$.
- $\mathsf{Verify}(r, \mathsf{com}, i, b, \boldsymbol{\pi}_i)$: On input a string $r \in \{0,1\}^{\ell_{\mathsf{pub}}}$, a commitment $\mathsf{com} = (i_1, \ldots, i_N, z_1, \ldots, z_N, \mathbf{c})$, an index $i \in [\ell]$, a bit $b \in \{0,1\}$ and an opening $\boldsymbol{\pi}_i \in \mathbb{Z}_q^t$, the verification algorithm proceeds as follows:
  1. Let $(\mathsf{crs}_{\mathsf{samp}}, \tilde{\mathbf{v}}_1, \ldots, \tilde{\mathbf{v}}_\ell)$ be the tuple associated with $r$. Then compute $\mathbf{v}_i = \tilde{\mathbf{v}}_i + \sum_{j \in [N]} z_j \boldsymbol{\eta}_{i_j}$.
  2. If $\mathbf{c} = \perp$, then output 1 if $b = 0$ and output 0 if $b = 1$.
  3. Otherwise, expand $(\mathbf{A}_1, \ldots, \mathbf{A}_\ell, \mathsf{td}_{\mathsf{samp}}) \leftarrow \mathsf{Expand}(1^\lambda, 1^\ell, \mathsf{crs}_{\mathsf{samp}})$.

4. Output 1 if

$$\|\boldsymbol{\pi}_i\| \le B_{\mathsf{max}} \quad \text{and} \quad \mathbf{A}_i\boldsymbol{\pi}_i = \mathbf{c} \quad \text{and} \quad \mathbf{v}_i^\intercal\boldsymbol{\pi}_i \in \left[\lfloor q/2\rfloor \cdot b - B_{\mathsf{round}}, \lfloor q/2\rfloor \cdot b + B_{\mathsf{round}}\right].$$

Output 0 otherwise.

In addition, we define the (TrapCoin, Extract) algorithms for the $\mu$-extraction property as follows:

- $\mathsf{TrapCoin}(1^\lambda, 1^\ell)$: On input a security parameter $\lambda$ and a length parameter $\ell$, the trapdoor public-coin generation algorithm proceeds as follows:
  1. Sample $r_{\mathsf{samp}} \xleftarrow{\mathrm{R}} \{0,1\}^{\ell_{\mathsf{samp}}}$ and let $\mathsf{crs}_{\mathsf{samp}} = \mathsf{Gen}(1^\lambda, 1^\ell; r_{\mathsf{samp}})$. Then expand $\mathsf{crs}_{\mathsf{samp}}$ by computing $(\mathbf{A}_1, \ldots, \mathbf{A}_\ell, \mathsf{td}_{\mathsf{samp}}) = \mathsf{Expand}(1^\lambda, 1^\ell, \mathsf{crs}_{\mathsf{samp}})$.
  2. Sample $i'_1, \ldots, i'_N \xleftarrow{\mathrm{R}} [t]$ and $z'_1, \ldots, z'_N \xleftarrow{\mathrm{R}} \mathbb{Z}_q$.
  3. Sample $\mathbf{s}_1, \ldots, \mathbf{s}_\ell \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$, $\mathbf{e}_1, \ldots, \mathbf{e}_\ell \leftarrow D^t_{\mathbb{Z}, s_{\mathsf{LWE}}}$. Set $\mathbf{v}_i^\intercal = \mathbf{s}_i^\intercal\mathbf{A}_i + \mathbf{e}_i^\intercal$ for $i \in [\ell]$.
  4. Compute $\tilde{\mathbf{v}}_i = \mathbf{v}_i - \sum_{j\in[N]} z'_j\boldsymbol{\eta}_{i'_j}$ for $i \in [\ell]$. Write $\tilde{\mathbf{v}}_i = [\tilde{v}_{i,1}, \ldots, \tilde{v}_{i,t}]$.
  5. For $i \in [\ell]$ and $j \in [t]$, sample $\gamma_{i,j} \xleftarrow{\mathrm{R}} [0, \lfloor 2^{\lceil \log q\rceil+\lambda}/q\rfloor - 1]$, and set $r_{i,j} = q\gamma_{i,j} + \tilde{v}_{i,j}$, represented as a binary string in $\{0,1\}^{\lceil \log q\rceil+\lambda}$. Finally, let $r_i = r_{i,1}\| \cdots \|r_{i,t} \in \{0,1\}^{t(\lceil \log q\rceil+\lambda)}$.
  6. Output $r^* = r_{\mathsf{samp}}\|r_1\| \cdots \|r_\ell$ and $\mathsf{td} = (r^*, i'_1, \ldots, i'_N, z'_1, \ldots, z'_N, \mathbf{s}_1, \ldots, \mathbf{s}_\ell)$.
- $\mathsf{Extract}(\mathsf{td}, \mathsf{com})$: On input a trapdoor $\mathsf{td} = (r^*, i'_1, \ldots, i'_N, z'_1, \ldots, z'_N, \mathbf{s}_1, \ldots, \mathbf{s}_\ell)$ and a commitment $\mathsf{com} = (i_1, \ldots, i_N, z_1, \ldots, z_N, \mathbf{c})$, the extraction algorithm does the following:
  1. If $i'_k \neq i_k$ or $z'_k \neq z_k$ for any $k \in [N]$, abort.
  2. Output $\boldsymbol{\rho}$ where $\rho_i \leftarrow \lfloor \mathbf{s}_i^\intercal\mathbf{c}\rceil$ for $i \in [\ell]$.

**Theorem 4.2** (Correctness). *If $\Pi_{\mathsf{samp}}$ is correct, then Construction 4.1 satisfies correctness.*

*Proof.* Take any $\lambda, \ell \in \mathbb{N}$. Take any public coin $r \in \{0,1\}^{\ell_{\mathsf{pub}}}$ and let $(\mathsf{crs}_{\mathsf{samp}}, \tilde{\mathbf{v}}_1, \ldots, \tilde{\mathbf{v}}_\ell)$ be the tuple associated with $r$. Take any commitment $\mathsf{com} = (i_1, \ldots, i_N, z_1, \ldots, z_N, \mathbf{c})$, openings $\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_\ell$ and string $\boldsymbol{\rho} = \rho_1 \cdots \rho_\ell$ in the support of $\mathsf{GenBits}(1^\lambda, 1^\ell, r)$. Consider the two possibilities:

- If $\mathbf{c} = \perp$ then by construction of GenBits, we have $\boldsymbol{\rho} = \mathbf{0}^\ell$. By construction, we have $\mathsf{Verify}(r, \mathsf{com}, i, 0, \boldsymbol{\pi}_i) = 1$ for all $i \in [\ell]$.
- Otherwise suppose $\mathbf{c} \in \mathbb{Z}_q^n$. By construction, this means that GenBits sampled $\mathbf{w}_1, \ldots, \mathbf{w}_\ell \xleftarrow{\mathrm{R}} \{0,1\}^t$ and computed $(\boldsymbol{\pi}'_1, \ldots, \boldsymbol{\pi}'_\ell, \mathbf{c}) \leftarrow \mathsf{SampleMultPre}(\mathsf{td}_{\mathsf{samp}}, \mathbf{A}_1\mathbf{w}_1, \ldots, \mathbf{A}_\ell\mathbf{w}_\ell)$. By correctness of $\Pi_{\mathsf{samp}}$, this means that for all $i \in [\ell]$, we have $\mathbf{A}_i\boldsymbol{\pi}'_i = \mathbf{c} + \mathbf{A}_i\mathbf{w}_i$. Next, GenBits sets $\boldsymbol{\pi}_i = \boldsymbol{\pi}'_i - \mathbf{w}_i$, so for all $i \in [\ell]$, we have

$$\mathbf{A}_i\boldsymbol{\pi}_i = \mathbf{A}_i(\boldsymbol{\pi}'_i - \mathbf{w}_i) = \mathbf{A}_i\boldsymbol{\pi}'_i - \mathbf{A}_i\mathbf{w}_i = \mathbf{c} + \mathbf{A}_i\mathbf{w}_i - \mathbf{A}_i\mathbf{w}_i = \mathbf{c}.$$

  Finally, by the bound checks in GenBits, it outputs $\mathbf{c} \in \mathbb{Z}_q^n$ only if

$$\|\boldsymbol{\pi}_i\| \le B_{\mathsf{max}} \quad \text{and} \quad \mathbf{v}_i^\intercal\boldsymbol{\pi}_i \in \left[\lfloor q/2\rfloor\, \rho_i - B_{\mathsf{round}}, \lfloor q/2\rfloor\, \rho_i + B_{\mathsf{round}}\right].$$

  In this case, all of the verification checks pass and $\mathsf{Verify}(r, \mathsf{com}, i, \rho_i, \boldsymbol{\pi}_i) = 1$. □

**Theorem 4.3** (Succinctness). *If there exists a polynomial $\mathsf{poly}(\cdot,\cdot)$ such that $N, t, \log q \le \mathsf{poly}(\lambda, \log \ell)$, then Construction 4.1 is succinct.*

*Proof.* Take any $r \in \{0,1\}^{\ell_{\mathsf{pub}}}$ and any $(\mathsf{com}, \boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_\ell, \boldsymbol{\rho})$ in the support of $\mathsf{GenBits}(1^\lambda, 1^\ell, r)$, where $\mathsf{com} = (i_1, \ldots, i_N, z_1, \ldots, z_N, \mathbf{c})$ for $i_j \in [t]$, $z_j \in \mathbb{Z}_q$, and $\mathbf{c} \in \mathbb{Z}_q^t$. Then, com can be represented in $N(\lceil \log t\rceil + \lceil \log q\rceil) + t\lceil \log q\rceil$ bits, as required. □

## 4.1 Security Analysis of Construction 4.1

In this section, we give the security analysis for Construction 4.1. In the following section, we show how to instantiate the underlying lattice parameters to obtain our main result (Theorem 1.1).

### 4.1.1 Mode Indistinguishability

In this section, we prove that Construction 4.1 satisfies mode indistinguishability (Definition 3.1). While Theorem 3.2 only requires normal mode indistinguishability, our proof of $\mu$-extractability (see Theorem 4.9 in Section 4.1.2) requires quasi-polynomial mode indistinguishability. For this reason, we rely on quasi-polynomially-secure LWE.

**Theorem 4.4** (Mode Indistinguishability). *Suppose c-quasi-polynomial* $\mathsf{LWE}_{n,t,q,s_{\mathsf{LWE}}}$ *holds with constant* $c > 1$ *and* $\Pi_{\mathsf{samp}}$ *has a transparent setup and is perfectly somewhere programmable. Then, Construction 4.1 satisfies c-quasi-polynomial mode indistinguishability.*

*Proof.* Let $\mathcal{A}$ be an efficient mode indistinguishability adversary. Next, $\Pi_{\mathsf{samp}}$ satisfies perfect somewhere programmability, so let GenProg be the associated algorithm. For each $k \in [\ell+1]$, we define a sequence of hybrid experiments between a challenger and $\mathcal{A}$ as follows:

- $\mathsf{Hyb}_0$: This is the mode indistinguishability game where the challenger samples a uniform random public-coin. Namely, the experiment proceeds as follows:
  1. The challenger samples $r^* \xleftarrow{R} \{0,1\}^{\ell_{\mathsf{pub}}}$.
  2. The challenger gives $(1^\lambda, 1^\ell, r^*)$ to $\mathcal{A}$. Algorithm $\mathcal{A}$ outputs a bit $b \in \{0,1\}$, which is the output of the experiment.
- $\mathsf{Hyb}_{k,1}$: This is the mode indistinguishability game, except the challenger samples the vectors $\tilde{\mathbf{v}}_1, \ldots, \tilde{\mathbf{v}}_{k-1}$ as LWE instances and then reverse samples the public-coin $r^* \in \{0,1\}^{\ell_{\mathsf{pub}}}$. Specifically, the experiment proceeds as follows:
  1. The challenger starts by sampling $r^*_{\mathsf{samp}} \xleftarrow{R} \{0,1\}^{\ell_{\mathsf{samp}}}$ and sets $\mathsf{crs}_{\mathsf{samp}} \leftarrow \mathsf{Gen}(1^\lambda, 1^\ell; r^*_{\mathsf{samp}})$. Then it computes $(\mathbf{A}_1, \ldots, \mathbf{A}_\ell, \mathsf{td}_{\mathsf{samp}}) = \mathsf{Expand}(1^\lambda, 1^\ell, \mathsf{crs}_{\mathsf{samp}})$.
  2. Next, the challenger samples $i'_1, \ldots, i'_N \xleftarrow{R} [t]$ and $z'_1, \ldots, z'_N \xleftarrow{R} \mathbb{Z}_q$.
  3. For each $i \in [\ell]$, the challenger proceeds as follows:
     - If $i < k$, the challenger samples $\mathbf{s}_i \xleftarrow{R} \mathbb{Z}_q^n$, $\mathbf{e}_i \leftarrow D^t_{\mathbb{Z},s_{\mathsf{LWE}}}$ and sets $\mathbf{v}_i^\intercal = \mathbf{s}_i^\intercal \mathbf{A}_i + \mathbf{e}_i^\intercal$.
     - If $i \geq k$, sample $\mathbf{v}_i \xleftarrow{R} \mathbb{Z}_q^t$.

     For each $i \in [\ell]$, the challenger sets $\tilde{\mathbf{v}}_i = \mathbf{v}_i - \sum_{j \in [N]} z'_j \boldsymbol{\eta}_{i'_j}$.
  4. For all $i \in [\ell]$ and $j \in [t]$:
     (a) The challenger samples $\gamma_{i,j} \xleftarrow{R} [0, \lfloor 2^{\lceil \log q \rceil + \lambda}/q \rfloor - 1]$.
     (b) The challenger sets $r^*_{i,j} = q\gamma_{i,j} + \tilde{v}_{i,j}$, represented as a binary string in $\{0,1\}^{\lceil \log q \rceil + \lambda}$.

     Finally, the challenger sets $r^*_i = r^*_{i,1} \| \cdots \| r^*_{i,t} \in \{0,1\}^{t(\lceil \log q \rceil + \lambda)}$.
  5. The challenger sets $r^* = r^*_{\mathsf{samp}} \| r^*_1 \| \cdots \| r^*_\ell \in \{0,1\}^{\ell_{\mathsf{samp}} + \ell t(\lceil \log q \rceil + \lambda)}$ and gives $(1^\lambda, 1^\ell, r^*)$ to $\mathcal{A}$. Algorithm $\mathcal{A}$ outputs a bit $b \in \{0,1\}$, which is the output of the experiment.
- $\mathsf{Hyb}_{k,2}$: Same as $\mathsf{Hyb}_{k,1}$, except the challenger samples $\mathbf{A}'_k \xleftarrow{R} \mathbb{Z}_q^{n \times t}$ and $\mathsf{crs}_{\mathsf{samp}} \leftarrow \mathsf{GenProg}(1^\lambda, 1^\ell, k, \mathbf{A}'_k)$. Let $r^* = \mathsf{crs}_{\mathsf{samp}} \| r^*_1 \| \cdots \| r^*_\ell$.
- $\mathsf{Hyb}_{k,3}$: Same as $\mathsf{Hyb}_{k,2}$, except the challenger samples $\mathbf{s}_k \xleftarrow{R} \mathbb{Z}_q^n$, $\mathbf{e}_k \leftarrow D^t_{\mathbb{Z},s_{\mathsf{LWE}}}$ and sets $\mathbf{v}_k^\intercal = \mathbf{s}_k^\intercal \mathbf{A}_k + \mathbf{e}_k^\intercal$.

For a hybrid experiment Hyb, we write $\mathsf{Hyb}(\mathcal{A})$ to denote the random variable corresponding to the output of an execution of Hyb with adversary $\mathcal{A}$.

**Claim 4.5.** *For every constant $c > 1$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that for all $\lambda \in \mathbb{N}$,*

$$\left|\Pr[\mathsf{Hyb}_{1,1}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_0(\mathcal{A}) = 1]\right| \leq 2^{-\log^c(\lambda)} \cdot \mathsf{negl}(\lambda).$$

*Proof.* Consider the distribution of $r^* = r^*_{\mathsf{samp}} \| r^*_1 \| \cdots \| r^*_\ell$ in the two experiments. First, the distribution of $r^*_{\mathsf{samp}}$ in the two experiments is identical, so it suffices to consider the distribution of $r^*_i$ for $i \in [\ell]$. In $\mathsf{Hyb}_0$, each $r^*_i$ is uniform over $\{0,1\}^{t(\lceil \log q \rceil + \lambda)}$. We consider the distribution in $\mathsf{Hyb}_{1,1}$:

- In $\mathsf{Hyb}_{1,1}$, the challenger samples $\mathbf{v}_i \xleftarrow{\mathrm{R}} \mathbb{Z}_q^t$. Since $\mathbf{v}_i$ is independent of $z'_1, \ldots, z'_N$, this means $\tilde{\mathbf{v}}_i = \mathbf{v}_i - \sum_{j \in [N]} z'_j \boldsymbol{\eta}_{i'_j}$ remains uniform over $\mathbb{Z}_q^t$.
- We conclude that each value $r_{i,j} = q\gamma_{i,j} + \tilde{v}_{i,j}$ is uniform over the set $[0, \lfloor 2^{\lceil \log q \rceil + \lambda}/q \rfloor \cdot q - 1]$. The statistical distance between this distribution and the uniform distribution over $[0, 2^{\lceil \log q \rceil + \lambda}]$ is at most $q/2^{\lceil \log q \rceil + \lambda} \leq 2^{-\lambda}$.
- The statistical distance between the uniform distribution over $\{0,1\}^{t(\lceil \log q \rceil + \lambda)}$ and $r^*_i$ is then at most $t \cdot 2^{-\lambda}$.

We conclude that the statistical distance between the distribution of $r^*$ in $\mathsf{Hyb}_{1,1}$ and the uniform distribution over $\{0,1\}^{\ell_{\mathsf{pub}}}$ is at most $\ell t \cdot 2^{-\lambda}$. This expression can be rewritten as $2^{-\log^c(\lambda)} \cdot 2^{-(\lambda - \log^c(\lambda))} \cdot \ell t$, which is $2^{-\log^c(\lambda)} \cdot \mathsf{negl}(\lambda)$ since $t, \ell = \mathsf{poly}(\lambda)$ and $2^{-(\lambda - \log^c(\lambda))}$ is negligible. □

**Claim 4.6.** *Suppose $\Pi_{\mathsf{samp}}$ has a transparent setup and is perfectly somewhere programmable. Then, for all $k \in [\ell]$,*

$$\Pr[\mathsf{Hyb}_{k,2}(\mathcal{A}) = 1] = \Pr[\mathsf{Hyb}_{k,1}(\mathcal{A}) = 1].$$

*Proof.* Note that in any execution of $\mathsf{Hyb}_{k,1}$, $r^*_{\mathsf{samp}} = \mathsf{crs}_{\mathsf{samp}}$ by transparent setup of $\Pi_{\mathsf{samp}}$. Since $\Pi_{\mathsf{samp}}$ is perfectly somewhere programmable, the distributions

$$\left\{\mathsf{crs} : \mathsf{crs} \leftarrow \mathsf{Gen}(1^\lambda, 1^\ell)\right\} \quad \text{and} \quad \left\{\widetilde{\mathsf{crs}} : \begin{array}{c} \mathbf{A}_k \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times t} \\ \widetilde{\mathsf{crs}} \leftarrow \mathsf{GenProg}(1^\lambda, 1^\ell, k, \mathbf{A}_k) \end{array}\right\}$$

are identically distributed for all $k \in [\ell]$. Thus, $\mathsf{Hyb}_{k,1}$ and $\mathsf{Hyb}_{k,2}$ are identical distributions and the claim follows. □

**Claim 4.7.** *Suppose quasi-polynomial $\mathsf{LWE}_{n,t,q,s_{\mathsf{LWE}}}$ holds with constant $c > 1$. Then, there exists a negligible function $\mathsf{negl}(\lambda)$ where for all $\lambda \in \mathbb{N}$,*

$$\sum_{k \in [\ell]} \left(\Pr[\mathsf{Hyb}_{k,3}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{k,2}(\mathcal{A}) = 1]\right) = 2^{-\log^c(\lambda)} \cdot \mathsf{negl}(\lambda).$$

*Proof.* We define an efficient adversary $\mathcal{B}$ in the $\mathsf{LWE}_{n,t,q,s_{\mathsf{LWE}}}$ game.

1. At the beginning of the game, algorithm $\mathcal{B}$ receives the LWE challenge $(\mathbf{A}^*, \mathbf{v}^*)$ from the challenger.
2. Algorithm $\mathcal{B}$ samples an index $k^* \xleftarrow{\mathrm{R}} [\ell]$ and sets $\mathbf{A}_{k^*} = \mathbf{A}^*$. Then, it samples $\mathsf{crs}_{\mathsf{samp}} \leftarrow \mathsf{GenProg}(1^\lambda, 1^\ell, k^*, \mathbf{A}_{k^*})$ and expands $(\mathbf{A}_1, \ldots, \mathbf{A}_\ell, \mathsf{td}_{\mathsf{samp}}) = \mathsf{Expand}(1^\lambda, 1^\ell, \mathsf{crs}_{\mathsf{samp}})$.
3. Algorithm $\mathcal{B}$ samples $i'_1, \ldots, i'_N \xleftarrow{\mathrm{R}} [t]$ and $z'_1, \ldots, z'_N \xleftarrow{\mathrm{R}} \mathbb{Z}_q$. For each $i \in [\ell]$, algorithm $\mathcal{B}$ sets $\mathbf{v}_i$ as follows:
   - If $i < k$, algorithm $\mathcal{B}$ samples $\mathbf{s}_i \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n$, $\mathbf{e}_i \leftarrow D^t_{\mathbb{Z}, s_{\mathsf{LWE}}}$ and sets $\mathbf{v}_i^\intercal = \mathbf{s}_i^\intercal \mathbf{A}_i + \mathbf{e}_i^\intercal$.
   - If $i = k^*$, algorithm $\mathcal{B}$ sets $\mathbf{v}_{k^*} = \mathbf{v}^*$.
   - If $i > k^*$, algorithm $\mathcal{B}$ samples $\mathbf{v}_i \xleftarrow{\mathrm{R}} \mathbb{Z}_q^t$.

   For each $i \in [\ell]$, algorithm $\mathcal{B}$ sets $\tilde{\mathbf{v}}_i = \mathbf{v}_i - \sum_{j \in [N]} z'_j \boldsymbol{\eta}_{i'_j}$.
4. For $i \in [\ell]$ and $j \in [t]$:
   (a) Algorithm $\mathcal{B}$ samples $\gamma_{i,j} \xleftarrow{\mathrm{R}} [0, \lfloor 2^{\lceil \log q \rceil + \lambda}/q \rfloor - 1]$.

(b) Algorithm $\mathcal{B}$ sets $r_{i,j} = q\gamma_{i,j} + \tilde{\mathbf{v}}_i[j]$, represented as a binary string of length $\lceil \log q \rceil + \lambda$.

Finally, algorithm $\mathcal{B}$ sets $r_i^* = r_{i,1}^* \| \cdots \| r_{i,t}^* \in \{0,1\}^{t(\lceil \log q \rceil + \lambda)}$.

5. Algorithm $\mathcal{B}$ sets $r^* = r_{\mathsf{samp}}^* \| r_1^* \| \cdots \| r_\ell^* \in \{0,1\}^{\ell_{\mathsf{samp}} + \ell t(\lceil \log q \rceil + \lambda)}$ and gives $(1^\lambda, 1^\ell, r^*)$ to $\mathcal{A}$. Finally, algorithm $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs.

Consider a run of $\mathcal{B}$ where $k^* = k$ for some fixed value $k \in [\ell]$. First, the LWE challenger always samples $\mathbf{A}^* \xleftarrow{R} \mathbb{Z}_q^{n \times m}$, so algorithm $\mathcal{B}$ correctly simulates the distribution of $\mathbf{A}_k$ according to the specification of $\mathsf{Hyb}_k$ or $\mathsf{Hyb}_{k+1}$. Consider now the distribution of $\mathbf{v}^*$:

- Suppose $\mathbf{v}_k = \mathbf{v}^* \xleftarrow{R} \mathbb{Z}_q^t$. In this case, algorithm $\mathcal{B}$ perfectly simulates an execution of $\mathsf{Hyb}_{k,2}$ for $\mathcal{A}$, and outputs 1 with probability $\Pr[\mathsf{Hyb}_{k,2}(\mathcal{A}) = 1]$.
- Suppose $\mathbf{v}_k = \mathbf{v}^* = \mathbf{s}^\intercal \mathbf{A}^* + \mathbf{e}^\intercal$ where $\mathbf{s} \xleftarrow{R} \mathbb{Z}_q^n$ and $\mathbf{e} \leftarrow D_{\mathbb{Z}, s_{\mathsf{LWE}}}^t$. Then, algorithm $\mathcal{B}$ perfectly simulates an execution of $\mathsf{Hyb}_{k,3}$ for $\mathcal{A}$ and outputs 1 with probability $\Pr[\mathsf{Hyb}_{k,3}(\mathcal{A}) = 1]$.

Finally, algorithm $\mathcal{B}$ samples $k^* \xleftarrow{R} [\ell]$, so its overall advantage is then

$$\frac{1}{\ell} \sum_{k \in [\ell]} (\Pr[\mathsf{Hyb}_{k,3}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{k,2}(\mathcal{A}) = 1]).$$

By $c$-quasi-polynomial hardness of $\mathsf{LWE}_{n,t,q,s_{\mathsf{LWE}}}$, this quantity is at most $2^{-\log^c(\lambda)} \cdot \mathsf{negl}'(\lambda)$ for some negligible function $\mathsf{negl}'(\cdot)$. Then,

$$\sum_{k \in [\ell]} (\Pr[\mathsf{Hyb}_{k,3}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{k,2}(\mathcal{A}) = 1]) \leq \ell \cdot 2^{-\log^c(\lambda)} \cdot \mathsf{negl}'(\lambda).$$

Since $\ell = \mathsf{poly}(\lambda)$, the claim holds. □

**Claim 4.8.** *Suppose $\Pi_{\mathsf{samp}}$ has a transparent setup and is perfectly somewhere programmable. Then, for all $k \in [\ell]$,*

$$\Pr[\mathsf{Hyb}_{k+1,1}(\mathcal{A}) = 1] = \Pr[\mathsf{Hyb}_{k,3}(\mathcal{A}) = 1].$$

*Proof.* Follows by the same argument as the proof of Claim 4.6. □

Returning now to the proof of Theorem 4.4, we have the following:

- $\mathsf{Hyb}_0$ is exactly the first mode indistinguishability experiment for algorithm $\mathcal{A}$.
- In $\mathsf{Hyb}_{\ell+1,1}$, every value $\mathbf{v}_i$ is sampled exactly as in TrapCoin. In this case, algorithm $\mathcal{B}$ perfectly simulates an execution of the second mode indistinguishability experiment for algorithm $\mathcal{A}$.

By Claims 4.6 to 4.8, we have

$$\begin{aligned}
&\left|\Pr[\mathsf{Hyb}_{\ell+1,1}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{1,1}(\mathcal{A}) = 1]\right| \\
&\qquad = \Bigg| \sum_{k \in [\ell]} (\Pr[\mathsf{Hyb}_{k+1,1}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{k,3}(\mathcal{A}) = 1]) \\
&\qquad\quad + \sum_{k \in [\ell]} (\Pr[\mathsf{Hyb}_{k,3}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{k,2}(\mathcal{A}) = 1]) \\
&\qquad\quad + \sum_{k \in [\ell]} (\Pr[\mathsf{Hyb}_{k,2}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{k,1}(\mathcal{A}) = 1]) \Bigg| \\
&\qquad = \left| \sum_{k \in [\ell]} (\Pr[\mathsf{Hyb}_{k,3}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{k,2}(\mathcal{A}) = 1]) \right| \\
&\qquad = 2^{-\log^c(\lambda)} \cdot \mathsf{negl}(\lambda)
\end{aligned}$$

for some negligible function $\mathsf{negl}(\cdot)$. By plugging in Claim 4.5, we get

$$\left|\Pr[\mathsf{Hyb}_{\ell+1,1}(\mathcal{A})=1]-\Pr[\mathsf{Hyb}_0(\mathcal{A})=1]\right| = 2^{-\log^c(\lambda)}\mathsf{negl}(\lambda)$$

and Theorem 4.4 holds. □

### 4.1.2 Extraction

In this section, we show that Construction 4.1 satisfies $\mu$-extractability for an inverse quasi-polynomial $\mu = 2^{-\Omega(\log^c \lambda)}$.

**Theorem 4.9** (Extraction). *Suppose c-quasi-polynomial* $\mathsf{LWE}_{n,t,q,s_{\mathsf{LWE}}}$ *holds with constant* $c > 4$ *and* $\Pi_{\mathsf{samp}}$ *is perfectly somewhere programmable. Suppose also* $B_{\mathsf{round}} + B_{\mathsf{max}}\sqrt{\log^c(\lambda)+\lambda}\cdot s_{\mathsf{LWE}} \le q/4$, *and* $N = O(\log^3\lambda)$. *Then, Construction 4.1 satisfies* $\mu$*-extraction where* $\mu = 2^{-\Omega(\log^c\lambda)}$.

*Proof.* Let $\mathcal{A}$ be an efficient adversary for the $\mu$-extraction game. We define a sequence of hybrid games between a challenger and $\mathcal{A}$ as follows:

- $\mathsf{Hyb}_1$: This is the $\mu$-extraction game where the output is 1 if event $E_\top$ occurs (i.e., the game where the adversary $\mathcal{A}$ does not abort). Specifically, the game proceeds as follows:

  1. The challenger starts by sampling $(r^*, \mathsf{td}) \leftarrow \mathsf{TrapCoin}(1^\lambda, 1^\ell)$. Namely, the challenger proceeds as follows:
     (a) Sample $r_{\mathsf{samp}} \xleftarrow{R} \{0,1\}^{\ell_{\mathsf{samp}}}$ and let $\mathsf{crs}_{\mathsf{samp}} = \mathsf{Gen}(1^\lambda, 1^\ell; r_{\mathsf{samp}})$. Then expand $\mathsf{crs}_{\mathsf{samp}}$ by computing $(\mathbf{A}_1, \ldots, \mathbf{A}_\ell, \mathsf{td}_{\mathsf{samp}}) = \mathsf{Expand}(1^\lambda, 1^\ell, \mathsf{crs}_{\mathsf{samp}})$.
     (b) Sample $i'_1, \ldots, i'_N \xleftarrow{R} [t]$ and $z'_1, \ldots, z'_N \xleftarrow{R} \mathbb{Z}_q$.
     (c) Sample $\mathbf{s}_1, \ldots, \mathbf{s}_\ell \xleftarrow{R} \mathbb{Z}_q^n$, $\mathbf{e}_1, \ldots, \mathbf{e}_\ell \leftarrow D^t_{\mathbb{Z}, s_{\mathsf{LWE}}}$. Set $\mathbf{v}_i^\top = \mathbf{s}_i^\top \mathbf{A}_i + \mathbf{e}_i^\top$ for $i \in [\ell]$.
     (d) Compute $\tilde{\mathbf{v}}_i = \mathbf{v}_i - \sum_{j\in[N]} z'_j \boldsymbol{\eta}_{i'_j}$ for $i \in [\ell]$.
     (e) For $i \in [\ell]$ and $j \in [t]$, sample $\gamma_{i,j} \xleftarrow{R} [0, \lfloor 2^{\lceil \log q\rceil+\lambda}/q\rfloor - 1]$ and set $r_{i,j} = q\gamma_{i,j} + \tilde{v}_{i,j}$, represented as a binary string in $\{0,1\}^{\lceil\log q\rceil+\lambda}$. Finally, let $r_i = r_{i,1}\|\cdots\|r_{i,t} \in \{0,1\}^{t(\lceil\log q\rceil+\lambda)}$.
     (f) Let $r^* = r_{\mathsf{samp}}\|r_1\|\cdots\|r_\ell$, $\mathsf{td} = (r^*, i'_1, \ldots, i'_N, z'_1, \ldots, z'_N, \mathbf{s}_1, \ldots, \mathbf{s}_\ell)$.

     The challenger sends $(1^\lambda, 1^\ell, r^*)$ to the adversary.
  2. The adversary $\mathcal{A}$ now either aborts with $\perp$ or outputs a set of indices $S \subseteq [\ell]$, a commitment $\mathsf{com} = (i_1, \ldots, i_N, z_1, \ldots, z_N, \mathbf{c})$ and openings $(\pi_i)_{i\in S}$.
  3. The output of the experiment is 1 if $\mathcal{A}$ does not abort. Otherwise, the output is 0.

- $\mathsf{Hyb}_2$: Same as $\mathsf{Hyb}_1$, except the challenger samples $\mathbf{v}_i \xleftarrow{R} \mathbb{Z}_q^t$ for all $i \in [\ell]$.
- $\mathsf{Hyb}_3$: Same as $\mathsf{Hyb}_2$, except the challenger additionally checks that $i'_k = i_k$ and $z'_k = z_k$ for all $k \in [N]$. If this condition does not hold, the output of the experiment is 0.
- $\mathsf{Hyb}_4$: Same as $\mathsf{Hyb}_3$, except the challenger reverts to setting $\mathbf{v}_i = \mathbf{s}_i^\top \mathbf{A}_i + \mathbf{e}_i^\top$ for all $i \in [\ell]$.
- $\mathsf{Hyb}_5$: Same as $\mathsf{Hyb}_4$, except the challenger aborts with output 0 if $\|\mathbf{e}_i\| > \sqrt{\log^c(\lambda)+\lambda}\cdot s_{\mathsf{LWE}}$ for any $i \in [\ell]$.
- $\mathsf{Hyb}_6$: Same as $\mathsf{Hyb}_5$, except the challenger also aborts with output 0 if there exists an index $i \in S$ where $\mathsf{Verify}(r^*, \mathsf{com}, i, 1-\rho_i, \pi_i) = 1$ and $\rho_i = \lfloor \mathbf{s}_i^\top \mathbf{c}\rceil$.
- $\mathsf{Hyb}_7$: Same as $\mathsf{Hyb}_6$, except the challenger no longer checks if there exists an $i \in [\ell]$ where $\|\mathbf{e}_i\| > \sqrt{\log^c(\lambda)+\lambda}\cdot s_{\mathsf{LWE}}$. This is the $\mu$-extraction game where the challenger outputs 1 if event $E_\top \wedge E_{\mathsf{lose}}$ occurs.

**Claim 4.10.** *Suppose c-quasi-polynomial* $\mathsf{LWE}_{n,t,q,s_{\mathsf{LWE}}}$ *holds with constant* $c > 1$ *and* $\Pi_{\mathsf{samp}}$ *is perfectly somewhere programmable. Then, there exists a negligible function* $\mathsf{negl}(\lambda)$ *where*

$$\left|\Pr[\mathsf{Hyb}_2(\mathcal{A})=1]-\Pr[\mathsf{Hyb}_1(\mathcal{A})=1]\right| = 2^{-\log^c(\lambda)}\cdot\mathsf{negl}(\lambda).$$

*Proof.* By Theorem 4.4, under the given conditions, Construction 4.1 satisfies $c$-quasi-polynomial mode indistinguishability. We use algorithm $\mathcal{A}$ to construct an adversary $\mathcal{B}$ for the mode indistinguishability game:

1. Algorithm $\mathcal{B}$ receives $(1^\lambda, 1^\ell, r^*)$, and sends $(1^\lambda, 1^\ell, r^*)$ to the adversary $\mathcal{A}$.
2. Algorithm $\mathcal{A}$ either aborts or outputs a set of indices $S \subseteq [\ell]$, a commitment $\mathsf{com} = (i_1, \ldots, i_N, z_1, \ldots, z_N, \mathbf{c})$ and a tuple of openings $(\pi_i)_{i \in S}$.
3. Algorithm $\mathcal{B}$ outputs 1 if $\mathcal{A}$ does not abort. Otherwise, algorithm $\mathcal{B}$ outputs 0.

We consider the two cases depending on the behavior of the mode indistinguishability challenger:

- If the challenger samples $r^* \leftarrow \mathsf{TrapCoin}(1^\lambda, 1^\ell)$, algorithm $\mathcal{B}$ perfectly simulates an execution of $\mathsf{Hyb}_1$ for $\mathcal{A}$, and outputs 1 with probability $\Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1]$.
- If the challenger samples $r^* \xleftarrow{R} \{0,1\}^{\ell_{\mathsf{pub}}}$, then algorithm $\mathcal{B}$ perfectly simulates an execution of $\mathsf{Hyb}_2$ for $\mathcal{A}$, and outputs 1 with probability $\Pr[\mathsf{Hyb}_2(\mathcal{A}) = 1]$.

Thus, the advantage of $\mathcal{B}$ for the mode indistinguishability game is $\left|\Pr[\mathsf{Hyb}_2(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1]\right|$. By Theorem 4.4, we can bound this quantity by $2^{-\log^c(\lambda)} \cdot \mathsf{negl}(\lambda)$ for some negligible function $\mathsf{negl}(\cdot)$, as required. □

**Claim 4.11.** *It holds that* $\Pr[\mathsf{Hyb}_3(\mathcal{A}) = 1] = t^{-N} q^{-N} \cdot \Pr[\mathsf{Hyb}_2(\mathcal{A}) = 1]$.

*Proof.* In $\mathsf{Hyb}_2$ and $\mathsf{Hyb}_3$, the only quantities that depend on $i'_k$ and $z'_k$ are the values $\tilde{\mathbf{v}}_i$ for $i \in [\ell]$. The challenger sets $\tilde{\mathbf{v}}_i = \mathbf{v}_i - \sum_{j \in [N]} z'_j \boldsymbol{\eta}_{i'_j}$. Since each $\mathbf{v}_i$ is uniform random, the values of $i'_k$ and $z'_k$ are information-theoretically hidden from the adversary $\mathcal{A}$ for all $k \in [N]$. Thus, the values $i_1, \ldots, i_N$ and $z_1, \ldots, z_N$ chosen by the adversary are *independent* of the values of $i'_1, \ldots, i'_N$ and $z'_1, \ldots, z'_N$. Since the challenger samples $i'_j \xleftarrow{R} [t]$ and $z'_j \xleftarrow{R} \mathbb{Z}_q$ for all $j \in [N]$, we have that whenever $\mathcal{A}$ does not abort (i.e., event $E_\top$ occurs), the probability that $i'_k = i_k$ and $z'_k = z_k$ for all $k \in [N]$ is exactly $t^{-N} q^{-N}$. □

**Claim 4.12.** *There exists a negligible function* $\mathsf{negl}(\lambda)$ *where*

$$\left|\Pr[\mathsf{Hyb}_4(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_3(\mathcal{A}) = 1]\right| = 2^{-\log^c(\lambda)} \cdot \mathsf{negl}(\lambda).$$

*Proof.* Follows by the same argument as the proof of Claim 4.10. □

**Claim 4.13.** *There exists a negligible function* $\mathsf{negl}(\lambda)$ *where*

$$\left|\Pr[\mathsf{Hyb}_5(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_4(\mathcal{A}) = 1]\right| = 2^{-\log^c(\lambda)} \cdot \mathsf{negl}(\lambda).$$

*Proof.* The only difference between $\mathsf{Hyb}_5$ and $\mathsf{Hyb}_4$ is we require for all $i \in [\ell]$ that $\|\mathbf{e}_i\| \le \sqrt{\log^c(\lambda) + \lambda} \cdot s_{\mathsf{LWE}}$. By Lemma 2.6, for each $i \in [\ell]$, $\|\mathbf{e}_i\| \le \sqrt{\log^c(\lambda) + \lambda} \cdot s_{\mathsf{LWE}}$ with all but probability $2^{-\log^c(\lambda) - \lambda}$. By a union bound, this occurs for all $i \in [\ell]$ with all but probability $2^{-\log^c(\lambda) - \lambda} \cdot \ell = 2^{-\log^c(\lambda)}(2^{-\lambda}\ell)$. Since $\ell = \mathsf{poly}(\lambda)$, the claim holds. □

**Claim 4.14.** *Suppose* $B_{\mathsf{round}} + B_{\mathsf{max}} \sqrt{\log^c(\lambda) + \lambda} \cdot s_{\mathsf{LWE}} \le q/4$. *Then* $\Pr[\mathsf{Hyb}_6(\mathcal{A}) = 1] = \Pr[\mathsf{Hyb}_5(\mathcal{A}) = 1]$.

*Proof.* Consider a run of $\mathsf{Hyb}_5(\mathcal{A})$ outputting 1. In particular, we know $i'_k = i_k$ and $z'_k = z_k$ for every $k \in [N]$. Consequently, for all $i \in [\ell]$, the value $\tilde{\mathbf{v}}_i - \sum_{j \in [N]} z'_j \boldsymbol{\eta}_{i'_j}$ computed by the Verify algorithm is indeed equal to $\mathbf{v}_i$ where $\mathbf{v}_i^\top = \mathbf{s}_i^\top \mathbf{A}_i + \mathbf{e}_i$. Suppose for some $i \in S$ and $b \in \{0,1\}$, $\|\mathbf{e}_i\| \le \sqrt{\log^c(\lambda) + \lambda} \cdot s_{\mathsf{LWE}}$ and $\mathsf{Verify}(r^*, \mathsf{com}, i, b, \boldsymbol{\pi}_i) = 1$. Then, the following conditions hold:

$$\|\boldsymbol{\pi}_i\| \le B_{\mathsf{max}} \quad \text{and} \quad \mathbf{A}_i \boldsymbol{\pi}_i = \mathbf{c} \quad \text{and} \quad \mathbf{v}_i^\top \boldsymbol{\pi}_i \in \left[\, \lfloor q/2 \rfloor\, b - B_{\mathsf{round}}, \lfloor q/2 \rfloor\, b + B_{\mathsf{round}} \right].$$

Notice $\mathbf{v}_i^\top \boldsymbol{\pi}_i = (\mathbf{s}_i^\top \mathbf{A}_i + \mathbf{e}_i^\top)\boldsymbol{\pi}_i = \mathbf{s}_i^\top (\mathbf{A}_i \boldsymbol{\pi}_i) + \mathbf{e}_i^\top \boldsymbol{\pi}_i = \mathbf{s}_i^\top \mathbf{c} + \mathbf{e}_i^\top \boldsymbol{\pi}_i$. Since $\|\boldsymbol{\pi}_i\| \le B_{\mathsf{max}}$, then $\left|\mathbf{e}_i^\top \boldsymbol{\pi}_i\right| \le B_{\mathsf{max}} \sqrt{\log^c(\lambda) + \lambda} \cdot s_{\mathsf{LWE}} \le q/4 - B_{\mathsf{round}}$, so

$$\mathbf{s}_i^\top \mathbf{c} = \mathbf{v}_i^\top \boldsymbol{\pi}_i - \mathbf{e}_i^\top \boldsymbol{\pi}_i \in \left[\, \lfloor q/2 \rfloor\, b - q/4, \lfloor q/2 \rfloor\, b + q/4 \right].$$

Consequently, $\rho_i = \lfloor \mathbf{s}_i^\top \mathbf{c} \rceil = b$, and the claim follows. □

**Claim 4.15.** *There exists a negligible function* $\mathsf{negl}(\lambda)$ *where*

$$\left|\Pr[\mathsf{Hyb}_7(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_6(\mathcal{A}) = 1]\right| \leq 2^{-\log^c(\lambda)}\mathsf{negl}(\lambda).$$

*Proof.* Follows by the same argument as the proof of Claim 4.13. □

Since $t, q$ are polynomials in $\lambda$ and $N = O(\log^3 \lambda)$, then, $t^{-N}q^{-N} = 2^{O(\log^4 \lambda)}$ and $t^N q^N \cdot 2^{-\log^c(\lambda)} \leq 2^{-\Omega(\log^c \lambda)}$ for any $c > 4$. By Claims 4.10 to 4.15, we conclude $\Pr[\mathsf{Hyb}_7(\mathcal{A}) = 1] \geq 2^{-\Omega(\log^c \lambda)}(\Pr[\mathsf{Hyb}_1(\mathcal{A}) = 1] - \mathsf{negl}(\lambda))$ and the theorem follows. □

### 4.1.3 Statistical Single-Bit Hiding

In this section, we show that Construction 4.1 satisfies statistical single-bit hiding (Definition 3.4). To argue this, we first abstract out the main information-theoretic problem that we are interested in. Namely, we define a *substring min-entropy preservation* game which the adversary must win in order to break single-bit hiding. At a high level, the game states that if the challenger samples a uniform $\mathbf{w} \xleftarrow{R} \{0,1\}^t$, then a random substring of $\mathbf{w}$ also has high entropy even given some information about $\mathbf{w}$. We give the formal definition below:

**Definition 4.16** (Substring Min-Entropy Preservation)**.** Let $n, t, q$ be lattice parameters $k$ be a number of samples. For a bit $b \in \{0,1\}$ and an adversary $\mathcal{A}$, we define the following experiment $\mathsf{Exp}_b^{\mathsf{SMEP}}$:

- On input the security parameter $1^\lambda$, the adversary outputs a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times t}$ and a vector $\mathbf{v} \in \mathbb{Z}_q^t$.
- The challenger samples $\mathbf{w} \xleftarrow{R} \{0,1\}^t$ and a vector of indices $I \xleftarrow{R} [t]^k$ (with repetitions) where $I = (i_1, \ldots, i_k)$. The challenger samples $\mathbf{z} \xleftarrow{R} \mathbb{Z}_q^k$. If $b = 0$, the challenger computes $u_b = \mathbf{z}^\intercal \mathbf{w}[I]$. If $b = 1$, the challenger samples $u_b \xleftarrow{R} \mathbb{Z}_q$.
- The challenger gives the challenge $(\mathbf{A}\mathbf{w}, \mathbf{v}^\intercal \mathbf{w}, I, \mathbf{z}, u_b)$ to $\mathcal{A}$. Algorithm $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$, which is the output of the experiment.

For any adversary $\mathcal{A}$, the advantage in the substring min-entropy preservation game is defined as

$$\mathsf{Adv}^{\mathsf{SMEP}}_{n,q,t,k}(\mathcal{A}) = \left|\Pr\left[\mathsf{Exp}_0^{\mathsf{SMEP}}(\mathcal{A}) = 1\right] - \Pr\left[\mathsf{Exp}_1^{\mathsf{SMEP}}(\mathcal{A}) = 1\right]\right|.$$

**Lemma 4.17** (Min-Entropy Preservation Advantage)**.** *Let* $n, t, q$ *be lattice parameters such that* $t \geq (2n+8)\log q + 6\log^2 \lambda + 10$*, where* $\lambda$ *is the security parameter. Let* $k \geq 8\log^2 \lambda \log t + 8 \log q \log t$ *be the number of samples. Then, for all (possibly unbounded) adversaries* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$*,*

$$\mathsf{Adv}^{\mathsf{SMEP}}_{n,q,t,k}(\mathcal{A}) = \mathsf{negl}(\lambda).$$

*Proof.* Fix an unbounded adversary $\mathcal{A}$. Without loss of generality, we can assume that $\mathcal{A}$ is deterministic. Define the auxiliary view of the adversary as the random variable $\mathsf{aux}_{\mathcal{A}} := (\mathbf{A}, \mathbf{v}, \mathbf{A}\mathbf{w}, \mathbf{v}^\intercal \mathbf{w})$. Note that $\mathbf{A}\mathbf{w} \in \mathbb{Z}_q^n$ and $\mathbf{v}^\intercal \mathbf{w} \in \mathbb{Z}_q$. Thus, the pair $(\mathbf{A}\mathbf{w}, \mathbf{v}^\intercal \mathbf{w})$ can be described by a bit-string of length $(n+1)\log q$. Additionally, observe that $\mathbf{A}, \mathbf{v}$ are chosen independent of $\mathbf{w}$. Since $\mathbf{w} \xleftarrow{R} \{0,1\}^t$, we know that $\mathbf{H}_\infty(\mathbf{w}) = t$. By Lemma 2.1 and using the fact that the pair $(\mathbf{A}, \mathbf{v})$ is independent of $\mathbf{w}$, we have that

$$\widetilde{\mathbf{H}}_\infty(\mathbf{w} \mid \mathsf{aux}_{\mathcal{A}}) = \widetilde{\mathbf{H}}_\infty(\mathbf{w} \mid (\mathbf{A}\mathbf{w}, \mathbf{v}^\intercal \mathbf{w})) \geq t - (n+1)\log q.$$

Now, we appeal to Lemma 2.3 with $c = \log^2 \lambda + \log q$. To do so, we need to show that the following inequality holds:

$$\widetilde{\mathbf{H}}_\infty(\mathbf{w} \mid \mathsf{aux}_{\mathcal{A}}) \geq t - (n+1)\log q \geq 2\frac{\log^2 \lambda + \log q}{k} t(1 + \log t) + 3\log^2 \lambda + 3\log q + 5$$

We rewrite the last inequality as

$$t\left(1 - \frac{2}{k}\left(\log^2 \lambda + \log q + \log^2 \lambda \log t + \log q \log t\right)\right) \geq (n+4)\log q + 3\log^2 \lambda + 5$$

By taking $k \geq 8(\log^2 \lambda \log t + \log q \log t)$, we guarantee that

$$1 - \frac{2(\log^2 \lambda + \log q)}{k} - \frac{2(\log^2 \lambda \log t + \log q \log t)}{k} \geq \frac{1}{2}.$$

By taking $t \geq (2n+8)\log q + 6\log^2 \lambda + 10$, we guarantee that

$$\frac{t}{2} \geq (n+4)\log q + 3\log^2 \lambda + 5.$$

Thus, by Lemma 2.3, we get that $\widetilde{\mathbf{H}}_\infty(\mathbf{w}[I] \mid (\mathsf{aux}_{\mathcal{A}}, I)) \geq \log^2 \lambda + \log q$. Finally, we can apply Lemma 2.2 to get that the statistical distance between the following distribution is $\mathsf{negl}(\lambda)$:

$$\left\{ (\mathbf{z}^\mathsf{T}\mathbf{w}[I], \mathsf{aux}_{\mathcal{A}}, \mathbf{z}, I) : \begin{array}{c} \mathbf{w} \xleftarrow{\mathrm{R}} \{0,1\}^t \\ I \xleftarrow{\mathrm{R}} [t]^k \\ \mathbf{z} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^t \end{array} \right\} \quad \text{and} \quad \left\{ (u, \mathsf{aux}_{\mathcal{A}}, \mathbf{z}, I) : \begin{array}{c} \mathbf{w} \xleftarrow{\mathrm{R}} \{0,1\}^t \\ I \xleftarrow{\mathrm{R}} [t]^k \\ \mathbf{z} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^t \\ u \xleftarrow{\mathrm{R}} \mathbb{Z}_q \end{array} \right\}.$$

These distributions correspond to the view of $\mathcal{A}$ in $\mathsf{Exp}_0^{\mathsf{SMEP}}$ and $\mathsf{Exp}_1^{\mathsf{SMEP}}$, so the advantage of $\mathcal{A}$ in the substring min-entropy preservation game is at most $\mathsf{negl}(\lambda)$ and the claim follows. □

**Theorem 4.18** (Statistical Single-Bit Hiding)**.** *Suppose* $t \geq (2n+8)\log q + 6\log^2 \lambda + 10$ *and* $k \geq 8\log^2 \lambda \log t + 8 \log q \log t$ *and that* $\ell$ *is some polynomial in* $\lambda$*. Suppose also that* $B_{\mathsf{round}} \geq q/4 - q/(42\ell)$ *and* $B_{\mathsf{max}} \geq \sqrt{t\ell}\, s_{\mathsf{LWE}}$*. Then Construction 4.1 satisfies statistical single-bit hiding.*

*Proof.* We begin by defining a series of hybrids in order to show that the advantage of any unbounded adversary in the single-bit hiding game is negligible. Fix an unbounded adversary $\mathcal{A}$, and assume that $\mathcal{A}$ is deterministic without loss of generality. For any $d' \in [\lambda]$, define the hybrid experiment $\mathsf{Hyb}_{d'}$ as follows:

1. On input $(1^\lambda, 1^\ell)$, the adversary $\mathcal{A}$ sends a public-coin $r \in \{0,1\}^{\ell_{\mathsf{pub}}}$ and an index $i^* \in [\ell]$ to the challenger.
2. The challenger then proceeds as follows:
   (a) Let $(\mathsf{crs}_{\mathsf{samp}}, \tilde{\mathbf{v}}_1, \ldots, \tilde{\mathbf{v}}_\ell)$ be the tuple associated with the public-coin $r$. Then, the challenger expands $(\mathbf{A}_1, \ldots, \mathbf{A}_\ell, \mathsf{td}_{\mathsf{samp}}) = \mathsf{Expand}(1^\lambda, 1^\ell, \mathsf{crs}_{\mathsf{samp}})$.
   (b) Sample $i_1, \ldots, i_N \xleftarrow{\mathrm{R}} [t]$ and $z_1, \ldots, z_N \xleftarrow{\mathrm{R}} \mathbb{Z}_q$, and compute $\mathbf{v}_i = \tilde{\mathbf{v}}_i + \sum_{j \in [N]} z_j \boldsymbol{\eta}_{i_j}$.
   (c) Next, for $d \in [\lambda]$, the challenger proceeds as follows:
   - Sample $\mathbf{w}_{d,i} \xleftarrow{\mathrm{R}} \{0,1\}^t$ and compute $\mathbf{y}_{d,i} = \mathbf{A}_i \mathbf{w}_{d,i}$ for all $i \in [\ell]$.
   - Sample $(\boldsymbol{\pi}'_{d,1}, \ldots, \boldsymbol{\pi}'_{d,\ell}, \mathbf{c}_d) \leftarrow \mathsf{SampleMultPre}(\mathsf{td}_{\mathsf{samp}}, \mathbf{y}_{d,1}, \ldots, \mathbf{y}_{d,\ell})$.
   - For each $i \in [\ell]$, set $\boldsymbol{\pi}_{d,i} = \boldsymbol{\pi}'_{d,i} + \mathbf{w}_{d,i}$
   - For each $i \neq i^*$, set $u_{d,i} = \mathbf{v}_i^\mathsf{T} \boldsymbol{\pi}_{d,i}$.
   - If $d > d'$, set $u_{d,i^*} = \mathbf{v}_{i^*}^\mathsf{T} \boldsymbol{\pi}_{d,i^*}$. Otherwise if $d \leq d'$ then sample $u_{d,i^*} \xleftarrow{\mathrm{R}} \mathbb{Z}_q$.
   - For each $i \in [\ell]$, if $\|\boldsymbol{\pi}_{d,i}\| > B_{\mathsf{max}}$ then set $\rho_{d,i} = \perp$. Otherwise, set $\rho_{d,i}$ as follows:

$$\rho_{d,i} = \begin{cases} 0 & u_{d,i} \in [-B_{\mathsf{round}}, B_{\mathsf{round}}] \\ 1 & u_{d,i} \in \left[ \lfloor q/2 \rfloor - B_{\mathsf{round}}, \lfloor q/2 \rfloor + B_{\mathsf{round}} \right] \\ \perp & \text{otherwise.} \end{cases} \tag{4.1}$$

   (d) The challenger constructs the challenge as follows:
   - If for all $d \in [\lambda]$, there exists an index $i \in [\ell]$ where $\rho_{d,i} = \perp$, then the challenger sets $\mathbf{c} = \perp$ and $\rho_i = 0$ and $\boldsymbol{\pi}_i = \perp$ for all $i \in [\ell]$.
   - Otherwise, let $d^* \in [\lambda]$ be the first index where $\rho_{d^*,i} \in \{0,1\}$ for all $i \in [\ell]$. Then the challenger sets $\mathbf{c} = \mathbf{c}_{d^*}$, and $\rho_i = \rho_{d^*,i}$ and $\boldsymbol{\pi}_i = \boldsymbol{\pi}_{d^*,i}$ for all $i \in [\ell]$.

(e) The challenger sends $\mathsf{com} = (i_1, \ldots, i_N, z_1, \ldots, z_N, \mathbf{c})$ and $(\boldsymbol{\pi}_i, \rho_i)_{i \neq i^*}$ as well as $\beta = \rho_{i^*}$ to the adversary.

3. The adversary outputs a bit $b'$, which is the output of the experiment.

Observe that $\mathsf{Hyb}_0$ is identical to the single-bit hiding game with $b = 0$. On the other hand, $\mathsf{Hyb}_\lambda$ is *almost* identical to the single-bit hiding game with $b = 1$, but not exactly. We address this issue in the following section. For now, we prove that the adversary has negligible probability of distinguishing any two consecutive hybrids:

**Lemma 4.19.** *Suppose* $t \geq (2n+8)\log q + 6\log^2 \lambda + 10$ *and* $k \geq 8\log^2 \lambda \log t + 8 \log q \log t$. *For all polynomials* $\ell = \ell(\lambda)$, *and all (possibly unbounded) adversaries* $\mathcal{A}$, *there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for all* $d' \in [\lambda]$:

$$\left|\Pr[\mathsf{Hyb}_{d'}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{d'-1}(\mathcal{A}) = 1]\right| \leq \mathsf{negl}(\lambda).$$

*Proof.* Let $\mathcal{A}$ be an unbounded adversary for the single-bit hiding game and let $\ell(\lambda)$ be a polynomial. We construct an algorithm $\mathcal{B}$ for the substring min-entropy preservation game in Definition 4.16 as follows:

1. On input a security parameter $1^\lambda$, algorithm $\mathcal{B}$ starts by sampling $d^* \xleftarrow{R} [\lambda]$ and runs $\mathcal{A}$ on $(1^\lambda, 1^\ell)$ to gets a public-coin $r \in \{0,1\}^{\ell_{\mathsf{pub}}}$ and an index $i^* \in [\ell]$.
2. Algorithm $\mathcal{B}$ computes the tuple $(\mathsf{crs}_{\mathsf{samp}}, \tilde{\mathbf{v}}_1, \ldots, \tilde{\mathbf{v}}_\ell)$ associated with the public-coin $r$. Then, it expands the matrices $(\mathbf{A}_1, \ldots, \mathbf{A}_\ell, \mathsf{td}_{\mathsf{samp}}) \leftarrow \mathsf{Expand}(1^\lambda, 1^\ell, \mathsf{crs}_{\mathsf{samp}})$, and submits $(\mathbf{A}_{i^*}, \tilde{\mathbf{v}}_{i^*})$ to the challenger.
3. The challenger responds with $\mathbf{y}_{d^*,i^*} \in \mathbb{Z}_q^n$, $\alpha \in \mathbb{Z}_q$, $I \in [t]^N$, $\mathbf{z} \in \mathbb{Z}_q^N$ and a scalar $u \in \mathbb{Z}_q$.
4. Algorithm $\mathcal{B}$ sets $\mathbf{v}_i = \tilde{\mathbf{v}}_i + \sum_{j \in [N]} z_j \boldsymbol{\eta}_{i_j}$, where $I = (i_1, \ldots, i_N)$.
5. For all $d \neq d^*$, algorithm $\mathcal{B}$ samples $\mathbf{c}_d, \rho_{d,1}, \ldots, \rho_{d,\ell}$ and $\boldsymbol{\pi}_{d,1}, \ldots, \boldsymbol{\pi}_{d,\ell}$ according to the specification of $\mathsf{Hyb}_{d^*}$.
6. For $d = d^*$, algorithm $\mathcal{B}$ does the following:
   (a) For all $i \neq i^*$, it samples $\mathbf{w}_{d^*,i} \xleftarrow{R} \{0,1\}^t$ and computes $\mathbf{y}_{d^*,i} = \mathbf{A}_i \mathbf{w}_{d^*,i}$.
   (b) It samples $(\boldsymbol{\pi}'_{d^*,1}, \ldots, \boldsymbol{\pi}'_{d^*,\ell}, \mathbf{c}_{d^*}) \leftarrow \mathsf{SampleMultPre}(\mathsf{td}_{\mathsf{samp}}, \mathbf{y}_{d^*,1}, \ldots, \mathbf{y}_{d^*,\ell})$.
   (c) For each $i \neq i^*$, it sets $\boldsymbol{\pi}_{d^*,i} = \boldsymbol{\pi}'_{d^*,i} + \mathbf{w}_{d^*,i}$ and $u_{d^*,i} = \mathbf{v}_i^\intercal \boldsymbol{\pi}_{d^*,i}$.
   (d) it computes $u_{d^*,i^*} = \mathbf{v}_{i^*}^\intercal \boldsymbol{\pi}'_{d^*,i^*} + u + \alpha$.
   (e) Finally, for all $i \in [\ell]$, it computes $\rho_{d^*,i}$ from $u_{d^*,i}$ according to Eq. (4.1).

   Finally, algorithm $\mathcal{B}$ sets $\mathbf{c}, \boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_\ell, \rho_1, \ldots, \rho_\ell$ using the same procedure as in $\mathsf{Hyb}_{d'}$, and sends the commitment $\mathsf{com} = (i_1, \ldots, i_N, z_1, \ldots, z_N, \mathbf{c})$, the openings $(\boldsymbol{\pi}_i, \rho_i)_{i \neq i^*}$, and the challenge bit $\rho_{i^*}$ to the adversary.

Fix some $d' \in [\lambda]$ and let $\varepsilon = |\Pr[\mathsf{Hyb}_{d'}(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{d'-1}(\mathcal{A}) = 1]|$. Since $d^*$ is sampled uniformly, then $d' = d^*$ with probability $1/\lambda$. Assume this event happens. Note that since $z_1, \ldots, z_N \xleftarrow{R} \mathbb{Z}_q$ and $i_1, \ldots, i_N \xleftarrow{R} [t]$, then for all $d \neq d'$, algorithm $\mathcal{B}$ simulates $\mathbf{c}_d, \rho_{d,1}, \ldots, \rho_{d,\ell}$ and $\boldsymbol{\pi}_{d,1}, \ldots, \boldsymbol{\pi}_{d,\ell}$ according to the specification of $\mathsf{Hyb}_{d'}$ and $\mathsf{Hyb}_{d'-1}$. Moreover, for all $i \neq i^*$, we have that $y_{d',i}$ are also simulated according to the specification of $\mathsf{Hyb}_{d'}$ and $\mathsf{Hyb}_{d'-1}$. In addition, the challenger of $\mathsf{Exp}_b^{\mathsf{SMEP}}$ (from Definition 4.16) samples $\mathbf{w}_{d',i^*} \xleftarrow{R} \{0,1\}^t$ and sets $\mathbf{y}_{d',i^*} = \mathbf{A}_{i^*} \mathbf{w}_{d',i^*}$. This implies that $(\boldsymbol{\pi}'_{d',1}, \ldots, \boldsymbol{\pi}'_{d',\ell}, \mathbf{c}_{d'})$ is also sampled according to the specification of $\mathsf{Hyb}_{d'}$ and $\mathsf{Hyb}_{d'-1}$. Now we do a case analysis on the bit $b$ in the definition of the $\mathsf{Exp}_b^{\mathsf{SMEP}}$ experiment:

- If $b = 0$, then $u = \mathbf{z}^\intercal \mathbf{w}_{d',i^*}[I]$ (where $\mathbf{z} \xleftarrow{R} \mathbb{Z}_q^N$ is sampled by the challenger). Recall also that $\alpha = \tilde{\mathbf{v}}_{i^*}^\intercal \mathbf{w}_{d',i^*}$. Hence $u + \alpha = \mathbf{z}^\intercal \mathbf{w}_{d',i^*}[I] + \tilde{\mathbf{v}}_{i^*}^\intercal \mathbf{w}_{d',i^*} = \mathbf{v}_{i^*}^\intercal \mathbf{w}_{d',i^*}$, where the last equality follows from the fact that $\mathbf{v}_{i^*} = \tilde{\mathbf{v}}_{i^*} + \sum_{j \in [N]} z_j \boldsymbol{\eta}_{i_j}$. In total, $u_{d',i^*} = \mathbf{v}_{i^*}^\intercal \boldsymbol{\pi}'_{d',i^*} + u + \alpha = \mathbf{v}_{i^*}^\intercal (\boldsymbol{\pi}'_{i^*} + \mathbf{w}_{d',i^*})$. Therefore $u_{d,i^*}$ in this case is sampled according to the specification of $\mathsf{Hyb}_{d'-1}$.
- If $b = 1$, then $u \xleftarrow{R} \mathbb{Z}_q$, then $u_{d,i^*}$ is a uniform random variable over $\mathbb{Z}_q$ and is thus sampled according to the specification of $\mathsf{Hyb}_{d'}$.

From here, we conclude that the advantage of $\mathcal{B}$ in breaking the game from Definition 4.16 is at least $\varepsilon/\lambda$ and therefore $\varepsilon$ is negligible by Lemma 4.17. □

**Handling rejection sampling.** We would like to say now that $\mathsf{Hyb}_\lambda$ is identical to the single-bit hiding game with $b = 1$. However, the rejection sampling introduces a bad case: if the rejection sampling procedure fails in $\mathsf{Hyb}_\lambda$, then the challenger sets $\beta = 0$. Whereas if the rejection sampling fails in the single-bit hiding game with $b = 1$, then the challenger would still set $\beta \xleftarrow{R} \{0,1\}$. This is natural, as we cannot expect the construction to be hiding when the rejection sampling fails (causing GenBits to output $\boldsymbol{\rho} = \mathbf{0}$). To address this, we show that each iteration of the rejection sampling fails with constant probability (e.g., at most $1/5$). After $\lambda$ independent attempts, the probability that *every* iteration fails is then negligible. This suffice to show that $\mathsf{Hyb}_\lambda$ is negligibly close to the single-bit hiding game with $b = 1$.

**Lemma 4.20.** *Fix any $d' \in [\lambda]$ and any $i' \in [\ell]$. Let $\boldsymbol{\pi}_{d',i'}$ be distributed as in $\mathsf{Hyb}_\lambda$. If $B_{\mathsf{max}} \geq \sqrt{t\ell}\, s_{\mathsf{LWE}}$ and $\Pi_{\mathsf{samp}}$ satisfies the preimage norm bound property, then*

$$\Pr\left[\left\|\boldsymbol{\pi}_{d',i'}\right\| > B_{\mathsf{max}}\right] \leq \frac{1}{10\ell}.$$

*Proof.* This follows immediately from the preimage norm bound property (Definition 2.8) and the choice of $B_{\mathsf{max}}$. □

**Lemma 4.21.** *Suppose $t \geq (2n+8)\log q + 6\log^2\lambda + 10$ and $k \geq 8\log^2\lambda \log t + 8\log q \log t$. Fix any $d' \in [\lambda]$ and any $i' \in [\ell]$. Let $u_{d',i'}$ be distributed as in $\mathsf{Hyb}_\lambda$. If $B_{\mathsf{round}} \geq q/4 - q/(42\ell)$ then*

$$\Pr\left[u_{d',i'} \notin [-B_{\mathsf{round}}, B_{\mathsf{round}}] \ \wedge\ u_{d',i'} \notin \left[\lfloor q/2\rfloor - B_{\mathsf{round}}, \lfloor q/2\rfloor + B_{\mathsf{round}}\right]\right] \leq \frac{1}{10\ell}. \tag{4.2}$$

*Proof.* Fix $d' \in [\lambda]$ and $i' \in [\ell]$. Let $\varepsilon$ be the probability from Eq. (4.2). If $i' = i^*$ then $u_{d,i^*}$ is sampled uniformly from $\mathbb{Z}_q$, so

$$\varepsilon = 1 - \frac{4B_{\mathsf{round}}}{q} \leq \frac{1}{10\ell}$$

for our choice of $B_{\mathsf{round}}$. Now suppose $i' \neq i^*$. We construct an adversary $\mathcal{B}$ for the game from Definition 4.16 as follows:

1. On input a security parameter $1^\lambda$, algorithm $\mathcal{B}$ runs $\mathcal{A}$ on $(1^\lambda, 1^\ell)$ and gets a public-coin $r \in \{0,1\}^{\ell_{\mathsf{pub}}}$ and an index $i^* \in [\ell]$.
2. Algorithm $\mathcal{B}$ computes the tuple $(\mathsf{crs}_{\mathsf{samp}}, \tilde{\mathbf{v}}_1, \ldots, \tilde{\mathbf{v}}_\ell)$ associated with the public-coin $r$. Then, it expands the matrices $(\mathbf{A}_1, \ldots, \mathbf{A}_\ell, \mathsf{td}_{\mathsf{samp}}) \leftarrow \mathsf{Expand}(1^\lambda, 1^\ell, \mathsf{crs}_{\mathsf{samp}})$ and submits $(\mathbf{A}_{i^*}, \tilde{\mathbf{v}}_{i^*})$ to the challenger.
3. The challenger responds with $y_{d',i'} \in \mathbb{Z}_q^n$, $\alpha \in \mathbb{Z}_q$, $I \in [t]^N$, $\mathbf{z} \in \mathbb{Z}_q^N$ and a scalar $u \in \mathbb{Z}_q$.
4. Algorithm $\mathcal{B}$ now simulates the challenger of $\mathsf{Hyb}_\lambda$ in order to compute $u_{d',i'}$:
   (a) For all $i \neq i'$, it samples $\mathbf{w}_{d',i} \xleftarrow{R} \{0,1\}^t$ and computes $\mathbf{y}_{d',i} = \mathbf{A}_i \mathbf{w}_{d',i}$.
   (b) It samples $(\boldsymbol{\pi}'_{d',1}, \ldots, \boldsymbol{\pi}'_{d',\ell}, \mathbf{c}_{d',i}) \leftarrow \mathsf{SampleMultPre}(\mathsf{td}_{\mathsf{samp}}, \mathbf{y}_{d',1}, \ldots, \mathbf{y}_{d',\ell})$.
   (c) It sets $u_{d',i'} = \mathbf{v}_{i'}^\intercal \boldsymbol{\pi}'_{d',i} + u + \alpha$.
5. It outputs 1 if $u_{d',i'} \notin [-B_{\mathsf{round}}, B_{\mathsf{round}}]$ or $u_{d',i'} \notin \left[\lfloor q/2\rfloor - B_{\mathsf{round}}, \lfloor q/2\rfloor + B_{\mathsf{round}}\right]$ and outputs 0 otherwise.

We analyze what happens for both values of the bit $b$ in the experiment $\mathsf{Exp}_b^{\mathsf{SMEP}}$:

- If $b = 0$, then $u = \mathbf{z}^\intercal \mathbf{w}_{d',i'}[I]$ for some $\mathbf{w}_{d',i,} \xleftarrow{R} \{0,1\}^t$ and thus, $u_{d',i'}$ is sampled according to the specification of $\mathsf{Hyb}_\lambda$ (by the same argument in the proof of Lemma 4.19). In this case, algorithm $\mathcal{B}$ outputs 1 with probability $\varepsilon$.
- If $b = 1$, then $u \xleftarrow{R} \mathbb{Z}_q$ and thus, $u_{d',i'}$ is uniform in the view of $\mathcal{B}$. In this case, algorithm $\mathcal{B}$ outputs 1 with probability $1 - \frac{4B_{\mathsf{round}}}{q}$.

By Lemma 4.17, the probability that $\mathcal{B}$ outputs 1 is negligibly close. Thus,

$$\varepsilon \leq 1 - \frac{4B_{\mathsf{round}}}{q} + \mathsf{negl}(\lambda).$$

Since $\ell$ is a polynomial and by our choice of $B_{\mathsf{round}}$, we have

$$\varepsilon \leq 1 - \frac{4B_{\mathsf{round}}}{q} + \mathsf{negl}(\lambda) \leq \frac{4}{42\ell} + \mathsf{negl}(\lambda) \leq \frac{1}{10\ell}$$

and the claim follows. □

**Corollary 4.22.** *Suppose $t \geq (2n+8)\log q + 6\log^2\lambda + 10$ and $k \geq 8\log^2\lambda \log t + 8\log q \log t$ and that $\ell$ is some polynomial in $\lambda$. Suppose also that $B_{\mathsf{round}} \geq q/4 - q/(42\ell)$ and $B_{\mathsf{max}} \geq \sqrt{t\ell}s_{\mathsf{LWE}}$. Then the rejection sampling procedure used $\mathsf{Hyb}_\lambda$ succeeds with overwhelming probability.*

*Proof.* By Lemmas 4.20 and 4.21 and a union bound over all $i \in [\ell]$, each iteration in the rejection sampling fails with probability at most $1/5$. Since we have $\lambda$ independent iterations, with all but negligible probability, at least one iteration succeeds. □

**Proof of Theorem 4.18.** Fix an unbounded adversary $\mathcal{A}$ for the single-bit hiding game. By Lemma 4.19 and a standard hybrid argument,

$$\left|\Pr[\mathsf{Hyb}_\lambda(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_0(\mathcal{A}) = 1]\right| = \mathsf{negl}(\lambda).$$

As we observed before, $\mathsf{Hyb}_0$ is just the single-bit hiding game with $b = 0$. On the other hand, conditioned on the rejection sampling procedure not failing, $\mathsf{Hyb}_\lambda$ is identical the single-bit hiding game with $b = 1$. By Corollary 4.22, we know that the rejection sampling fails with negligible probability, therefore $\mathsf{Hyb}_\lambda$ is statistically close to the single-bit hiding game with $b = 1$. In total, the adversary $\mathcal{A}$ has only a negligible advantage in the single-bit hiding game and the theorem follows. □

## 4.2 Parameter Instantiation

Let $\lambda$ be a security parameter and $\ell$ be a length parameter. We now provide one possible instantiation of the parameters in Construction 4.1 to satisfy the requirements of Theorems 4.2, 4.4, 4.9 and 4.18. In the following, we assume that $\ell$ is some polynomial in $\lambda$, so $\log\ell = O(\log\lambda)$.

- We set $s_{\mathsf{LWE}} = \lambda^\delta$ for some constant $\delta > 0$ (to be determined later).
- We instantiate $\Pi_{\mathsf{samp}}$ using Theorem 2.10 with parameters $n = \lambda$ and $q = \lambda^2 \cdot \ell^2 \cdot s_{\mathsf{LWE}}^2 = \lambda^{2+2\delta}\ell^2 = \mathsf{poly}(\lambda)$. With this setting of parameter, we have that $t = \lambda \cdot \mathsf{polylog}(\lambda)$.
- Next, we set $N = 8\log^2\lambda\log t + 8\log q\log t = O(\log^3\lambda)$.
- We set the bounds to be $B_{\mathsf{round}}(\lambda,\ell) = q/4 - q/(42\ell) = \lambda^{2+\delta}\ell^2/4 - \lambda^{2+\delta}\ell/42$ and $B_{\mathsf{max}}(\lambda,\ell) = \sqrt{t\ell}s_{\mathsf{LWE}} = \sqrt{\ell}\lambda^{1/2+\delta} \cdot \mathsf{polylog}(\lambda,\ell)$.
- Finally, we choose $\delta > 0$ (recall that $s_{\mathsf{LWE}} = \lambda^\delta$) so that the $c$-quasi-polynomial $\mathsf{LWE}_{n,m,q,s_{\mathsf{LWE}}}$ assumption holds for some constant $c > 4$.

We briefly verify that these parameters satisfy the necessary requirements. We start with Theorem 4.18:

- Note that the setting of $t$ by Theorem 2.10 gives $t > 3\lambda\log q$ which satisfies the requirement. In addition, $N, B_{\mathsf{round}}, B_{\mathsf{max}}$ were set to satisfy Theorem 4.18.

We now consider the conditions for Theorem 4.9:

- We require $B_{\mathsf{round}} + B_{\mathsf{max}}\sqrt{\log^c(\lambda)+\lambda} \cdot s_{\mathsf{LWE}} \leq q/4$. That is, we want to show $B_{\mathsf{max}}\sqrt{\log^c(\lambda)+\lambda} \cdot s_{\mathsf{LWE}} \leq q/(42\ell) = \lambda^{2+2\delta} \cdot \ell/42$. This exactly follows since $B_{\mathsf{max}}(\lambda,\ell) = \sqrt{\ell}\lambda^{1/2+\delta} \cdot \mathsf{polylog}(\lambda,\ell)$.
- Additionally, we have $N = O(\log^3\lambda)$.

Together with Theorem 3.2, we obtain a ZAP for NP from quasi-polynomial hardness of LWE with a polynomial modulus-to-noise ratio. This yields Theorem 1.1.

## Acknowledgments

We would like to thank Elahe Sadeghi for useful discussions. Brent Waters is supported by NSF CNS-1908611, CNS-2318701, and a Simons Investigator award. David J. Wu is supported by NSF CNS-2140975, CNS-2318701, a Sloan Fellowship, a Microsoft Research Faculty Fellowship, a Google Research Scholar Award, and an Amazon Research Award.

## References

[ADW09] Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In *CRYPTO*, 2009.

[BCD+25] Pedro Branco, Arka Rai Choudhuri, Nico Döttling, Abhishek Jain, Giulio Malavolta, and Akshayaram Srinivasan. Black-box non-interactive zero knowledge from vector trapdoor hash. In *EUROCRYPT*, 2025.

[BFJ+20] Saikrishna Badrinarayanan, Rex Fernando, Aayush Jain, Dakshita Khurana, and Amit Sahai. Statistical ZAP arguments. In *EUROCRYPT*, 2020.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *STOC*, 1988.

[BKM20] Zvika Brakerski, Venkata Koppula, and Tamer Mour. NIZK from LPN and trapdoor hash via correlation intractability for approximable relations. In *CRYPTO*, 2020.

[BKP+24] Nir Bitansky, Chethan Kamath, Omer Paneth, Ron D. Rothblum, and Prashant Nalini Vasudevan. Batch proofs are statistically hiding. In *STOC*, 2024.

[BLV03] Boaz Barak, Yehuda Lindell, and Salil P. Vadhan. Lower bounds for non-black-box zero knowledge. In *FOCS*, 2003.

[BP15] Nir Bitansky and Omer Paneth. Zaps and non-interactive witness indistinguishability from indistinguishability obfuscation. In *TCC*, 2015.

[BWW24] Eli Bradley, Brent Waters, and David J. Wu. Batch arguments to NIZKs from one-way functions. In *TCC*, 2024.

[BY92] Mihir Bellare and Moti Yung. Certifying cryptographic tools: The case of trapdoor permutations. In *CRYPTO*, 1992.

[CCH+19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. Fiat-Shamir: from practice to theory. In *STOC*, 2019.

[CCRR18] Ran Canetti, Yilei Chen, Leonid Reyzin, and Ron D. Rothblum. Fiat-Shamir and correlation intractability from strong KDM-secure encryption. In *EUROCRYPT*, 2018.

[CHK03] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In *EUROCRYPT*, 2003.

[CJJQ23] Geoffroy Couteau, Abhishek Jain, Zhengzhong Jin, and Willy Quach. A note on non-interactive zero-knowledge from CDH. In *CRYPTO*, 2023.

[CKSU21] Geoffroy Couteau, Shuichi Katsumata, Elahe Sadeghi, and Bogdan Ursu. Statistical ZAPs from group-based assumptions. In *TCC*, 2021.

[CKU20] Geoffroy Couteau, Shuichi Katsumata, and Bogdan Ursu. Non-interactive zero-knowledge in pairing-free groups from weaker assumptions. In *EUROCRYPT*, 2020.

[CL18] Ran Canetti and Amit Lichtenberg. Certifying trapdoor permutations, revisited. In *TCC*, 2018.

[CW23] Jeffrey Champion and David J. Wu. Non-interactive zero-knowledge from non-interactive batch arguments. In *CRYPTO*, 2023.

[DJJ24] Quang Dao, Aayush Jain, and Zhengzhong Jin. Non-interactive zero-knowledge from LPN and MQ. In *CRYPTO*, 2024.

[DN00] Cynthia Dwork and Moni Naor. Zaps and their applications. In *FOCS*, 2000.

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam D. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1), 2008.

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *FOCS*, 1990.

[FS90] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *STOC*, 1990.

[GJJM20] Vipul Goyal, Abhishek Jain, Zhengzhong Jin, and Giulio Malavolta. Statistical Zaps and new oblivious transfer protocols. In *EUROCRYPT*, 2020.

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

[GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *J. Cryptol.*, 7(1), 1994.

[GOS06a] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive Zaps and new techniques for NIZK. In *CRYPTO*, 2006.

[GOS06b] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In *EUROCRYPT*, 2006.

[GR13] Oded Goldreich and Ron D. Rothblum. Enhancements of trapdoor permutations. *J. Cryptol.*, 26(3), 2013.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4), 1999.

[JJ21] Abhishek Jain and Zhengzhong Jin. Non-interactive zero knowledge from sub-exponential DDH. In *EUROCRYPT*, 2021.

[KKS18] Yael Tauman Kalai, Dakshita Khurana, and Amit Sahai. Statistical witness indistinguishability (and more) in two messages. In *EUROCRYPT*, 2018.

[KMY23] Fuyuki Kitagawa, Takahiro Matsuda, and Takashi Yamakawa. NIZK from snargs. *J. Cryptol.*, 36(2), 2023.

[LPWW20] Benoît Libert, Alain Passelègue, Hoeteck Wee, and David J. Wu. New constructions of statistical NIZKs: Dual-mode DV-NIZKs and more. In *EUROCRYPT*, 2020.

[LVW19] Alex Lombardi, Vinod Vaikuntanathan, and Daniel Wichs. 2-message publicly verifiable WI from (subexponential) LWE. *IACR Cryptol. ePrint Arch.*, 2019.

[LVW20] Alex Lombardi, Vinod Vaikuntanathan, and Daniel Wichs. Statistical ZAPR arguments from bilinear maps. In *EUROCRYPT*, 2020.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, 2012.

[NZ96] Noam Nisan and David Zuckerman. Randomness is linear in space. *J. Comput. Syst. Sci.*, 52(1), 1996.

[PS19] Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for NP from (plain) learning with errors. In *CRYPTO*, 2019.

[QRW19] Willy Quach, Ron D. Rothblum, and Daniel Wichs. Reusable designated-verifier nizks for all NP from CDH. In *EUROCRYPT*, 2019.

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

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

[Vad06] Salil P. Vadhan. An unconditional study of computational zero knowledge. *SIAM J. Comput.*, 36(4), 2006.

[Wat24] Brent Waters. A new approach for non-interactive zero-knowledge from learning with errors. In *STOC*, 2024.

[WWW25] Brent Waters, Hoeteck Wee, and David J. Wu. New techniques for preimage sampling: Improved NIZKs and more from LWE. In *EUROCRYPT*, 2025.