# Black-Box Non-Interactive Zero Knowledge from Vector Trapdoor Hash

Pedro Branco
Bocconi

Arka Rai Choudhuri
Nexus

Nico Döttling
CISPA

Abhishek Jain
NTT Research and JHU

Giulio Malavolta
Bocconi

Akshayaram Srinivasan
University of Toronto

**Abstract**

We present a new approach for constructing non-interactive zero-knowledge (NIZK) proof systems from *vector trapdoor hashing* (VTDH) – a generalization of trapdoor hashing [Döttling et al., Crypto'19]. Unlike prior applications of trapdoor hash to NIZKs, we use VTDH to realize the hidden bits model [Feige-Lapidot-Shamir, FOCS'90] leading to *black-box* constructions of NIZKs. This approach gives us the following new results:

- A *statistically-sound* NIZK proof system based on the hardness of decisional Diffie-Hellman (DDH) and learning parity with noise (LPN) over finite fields with inverse polynomial noise rate. This gives the first statistically sound NIZK proof system that is not based on either LWE, or bilinear maps, or factoring.
- A dual-mode NIZK satisfying statistical zero-knowledge in the common random string mode and statistical soundness in the common reference string mode assuming the hardness of learning with errors (LWE) with *polynomial* modulus-to-noise ratio. This gives the first *black-box* construction of such a dual-mode NIZK under LWE. This improves the recent work of Waters (STOC'24) which relied on LWE with super-polynomial modulus-to-noise ratio and required a setup phase with private coins.

The above constructions are black-box and satisfy single-theorem zero-knowledge property. Building on the works of Feige et al.(FOCS'90) and Fischlin and Rohrback (PKC'21), we upgrade these constructions (under the same assumptions) to satisfy multi-theorem zero-knowledge property at the expense of making non-black-box use of cryptography.

## Contents

## 1 Introduction

Zero-knowledge (ZK) proofs [GMR85] allow a prover to convince a verifier about the truthfulness of a claim without leaking any other information. ZK proofs were originally envisioned as interactive protocols where the prover and the verifier exchange multiple messages with each other. Subsequently, ZK proofs were studied in the more challenging non-interactive setting, where much like standard mathematical proofs, the prover publishes a single message that can be verified by anyone. To bypass trivial impossibility results in this setting, the prover and the verifier are given access to a common reference string (CRS) sampled from some distribution. This notion is referred to as *non-interactive zero knowledge* (NIZK) [DMP88, BFM88, FLS90] and is central to the popularity of ZK.

While interactive ZK proofs for NP are long known to exist from one-way functions [GMW87], NIZKs have proven to be significantly harder to construct even from number-theoretic assumptions.[1] The celebrated work of Feige, Lapidot and Shamir [FLS90] proposed the first general approach to constructing NIZKs. Their approach involves two steps: first, unconditionally-secure NIZKs are constructed in an idealized model referred to as the hidden bits model where the prover has access to a large private random string and can choose to selectively reveal certain bits of the string to the verifier. The second step involves a generic transformation from NIZKs in the hidden bits model to NIZKs in the CRS model using a cryptographic object called *hidden bits generator* (HBG) [QRW19].[2] In the same work, [FLS90] gave the first construction of HBG based on certified trapdoor permutations which can be instantiated from the RSA assumption. More than a decade later, [CHK03, GOS06b, GOS06a] proposed instantiations based on bilinear maps. Constructions of NIZKs from other standard assumptions known to imply public-key encryption, however, curiously remained elusive for another decade.

In the late 2010s, researchers started to propose a new generation of NIZKs via the framework of correlation intractable hash (CIH) [CGH04]. Informally speaking, CIH is a family of keyed hash functions that allow for securely instantiating the Fiat-Shamir paradigm [FS87] for round-collapsing interactive proofs into non-interactive proof systems. Over the last five years, this framework has proven to be remarkably fruitful [CCRR18, HL18, CCH$^+$19, PS19, CKU20, BKM20, JJ21, DJJ24] leading to the first constructions of NIZKs from the learning with errors (LWE) assumption [CCH$^+$19, PS19], the joint hardness of decisional Diffie-Hellman (DDH) and learning parity with noise (LPN) assumptions [BKM20], the sub-exponential DDH assumption [JJ21], and more recently, the joint hardness of LPN and multivariate quadratic assumption [DJJ24]. Overall, this line of work has significantly reduced the gaps in our understanding about the existence of NIZKs.

Conceptually, the above two approaches to NIZKs, namely, HBG and CIH, are quite different. The latter approach, so far, has required powerful homomorphism techniques for realizing NIZKs from LWE [CCH$^+$19, PS19] and sub-exponential DDH [JJ21], or the seemingly unique low-depth decryption property of LPN-based encryption [BKM20, DJJ24]. It is natural to ask whether such specialized techniques are necessary for constructing NIZKs (from the aforementioned assumptions), or whether it is possible to obtain simpler constructions. Furthermore, all NIZK constructions achieved via the CIH framework inherently require *non-black-box* use of cryptography.[3] This poses a signif-

[1]In the Random Oracle model, NIZKs are known to exist unconditionally.

[2]This notion was implicit in [FLS90], but was first formalized in [QRW19]. See also [GO93] for the related notion of invariant signatures.

[3]In particular, current constructions require computation of the decryption algorithm of a public-key encryption scheme inside another cryptographic primitive.

icant barrier to the practical viability of NIZK constructions obtained via this approach. The HBG approach, in contrast, does not seem to pose such a barrier. Thus, achieving new instantiations of HBG is well-motivated from both theoretical and practical perspective.

A recent beautiful work of Waters [Wat24] tackled this challenge and presented a new construction of HBG from LWE with super-polynomial modulus-to-noise ratio. This, in turn, yields a black-box construction of NIZK satisfying single-theorem zero-knowledge property from the same assumption. Notably, his construction does not rely on homomorphic encryption techniques, thus seemingly opening a new door to simpler constructions of NIZKs. However, his construction crucially relies on lattice trapdoors; as such, the generality of the underlying ideas towards constructing HBG from other assumptions or the necessity of lattice trapdoors remain unclear.

## 1.1 Our Results

We provide a general approach for realizing the hidden bits generator (HBG) via a new notion called *vector trapdoor hash* (VTDH). Vector trapdoor hash generalizes the previously studied notion of trapdoor hash (TDH) [DGI$^+$19]. Trapdoor hash was used in recent works [BKM20, JJ21, DJJ24] to build NIZKs via the correlation intractability framework. As discussed earlier, however, this framework inherently makes non-black-box use of cryptography. Our approach, in contrast, yields *black-box* constructions (in the single-theorem zero-knowledge setting).

We give two instantiations of our approach. Some of the properties achieved by our constructions (highlighted below) were not known previously even via non-black-box techniques.

1. **Construction from DDH+LPN.** We give the *first* construction of HBG based on the hardness of DDH and LPN over finite fields with inverse polynomial noise rate. This gives the first construction of NIZK (black-box or non-black-box) satisfying *statistical soundness* without relying on LWE, or bilinear maps, or factoring. In contrast, the construction of Brakerski et al. [BKM20] based on correlation intractability (and hence, non-black-box) could only achieve computational soundness. The single-theorem zero-knowledge version of our construction makes black-box use of cryptography. By relying on the transformation in [FLS90], we get a multi-theorem version at the expense of making non-black-box use of a PRG.

2. **Construction from LWE.** We give a new construction of HBG with a *transparent setup* phase based on LWE with *polynomial modulus-to-noise ratio*. The recent construction of Waters [Wat24] relies on LWE with super-polynomial modulus-to-noise ratio and a non-transparent setup that uses private coins. Our construction has a dual-mode property, yieldings NIZKs that achieve statistical single-theorem zero-knowledge property in the common random string mode and statistical soundness in the common reference string mode. This gives the first *black-box* construction of such a dual-mode NIZK, matching the non-black-box construction of Peikert and Shiehian [PS19] in terms of assumptions. An interesting aspect of our work is that the construction and the proof of security do not rely on lattice trapdoors [GPV08], which were crucially employed in [Wat24].

   The transformation in [FLS90] from single-theorem to multi-theorem ZK does not preserve statistical zero-knowledge property in the common random string model. Building on the work of Fischlin and Rohrback [FR21], we give a single-theorem to multi-theorem transformation in the common random string model that preserves statistical zero-knowledge property assuming LWE. As in the [FLS90] transformation, this too makes non-black-box use of cryptography.

To show the versatility of our framework, in Appendix B, we show that Libert et al.'s construction [LPWW20] of NIZK in the designated verifier setting can be viewed in the framework of vector trapdoor hash.

## 2 Technical Overview

In this section, we give an overview of our constructions of vector trapdoor hash (VTDH). We begin with giving an overview of this new notion and how it implies hiddent bits generator (HBG) in Section 2.1. In Section 2.2, we give our construction from LWE. This construction has the advantage of having a transparent setup and its security can be based on LWE with polynomial modulus-to-noise ratio. In Section 2.4, we provide another construction whose security relies on the DDH and the LPN assumptions.

### 2.1 Vector Trapdoor Hash

Let us first recall the standard definition of a trapdoor hash function [DGI+19]. It consists of a standard (keyed) hash function $\mathsf{Hash}(\mathsf{hk}, \mathbf{x}) \to h$, augmented with the following additional algorithms:

- A key generation algorithm $\mathsf{Gen}$ that generates an encoding key $\mathsf{ek}$ along with a trapdoor $\mathsf{td}$, on input a linear function $f$.
- An encoding algorithm $\mathsf{Encode}$ that uses the key $\mathsf{ek}$ to encode the input $\mathbf{x}$, returning an encoding $e$.
- A decoding algorithm $\mathsf{Decode}$ that, given the trapdoor $\mathsf{td}$ and the hash $h$ generates a decoding $d$.

The fundamental property of a trapdoor hash is that $e - d = f(\mathbf{x})$, and furthermore the encoding key should not leak any information about the function $f$.

**Vector Trapdoor Hash.** In this work we consider a generalization of trapdoor hash functions, that we refer to a *vector trapdoor hash* (VTDH) functions, with a few syntactical differences:

1. **Setup.** The setup phase takes as input the security parameter and another parameter $k$ which denotes the number of encoding keys to be produced. We will only consider encoding keys for the *zero* function. As a result, the setup phase will output the hashing key $\mathsf{hk}$ along with $k$ sets of encoding keys $\mathsf{ek}_1, \ldots, \mathsf{ek}_k$ and trapdoors $\mathsf{td}_1, \ldots, \mathsf{td}_k$.
2. **Local openings.** The hashing algorithm now takes in a block vector $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_k)$ as input. We require the hashing algorithm to output the digest $h$, along with $k$ local openings $\pi_i$, one for each block of the input $\mathbf{x}$. We remark that the idea of local opening used here is slightly different from the traditional notion which gives a short proof that the $i$-th block is equal to $\mathbf{x}_i$. We consider a more general notion where a valid local opening is anything that passes the following verification test.
3. **Verification.** We also consider a verification algorithm $\mathsf{Verify}$ that determines the validity of each local opening $\pi_i$, given the hash digest $h$ and the key $\mathsf{hk}$.

4. **Encoding and Decoding.** The encoding algorithm takes in an encoding key $\mathsf{ek}_i$ and the corresponding local opening $\pi_i$ and produces and encoding bit $e_i$. The decoding algorithm takes in the hash digest $h$ and a trapdoor $\mathsf{td}_i$ and outputs the decoding bit $d_i$. As we only consider the zero function, it follows that $e_i = d_i$ for each $i \in [k]$ if they are honestly computed.

Completeness states that, for a honestly generated hash and local proofs, the verification always succeeds. For succinctness, we require that the size of the hash digest to be a fixed polynomial in the security parameter $\lambda$. As for security, we formalize two properties:

1. **Hiding.** For any index $i^* \in [k]$, $\mathsf{Encode}(\mathsf{ek}_{i^*}, \pi_{i^*})$ is pseudorandom even given all other local openings $\{\pi_j\}_{j \neq i^*}$.

2. **Statistical binding.** No adversary (even an unbounded one) should be able to compute a hash $h$, along with local openings $\pi_1, \ldots, \pi_k$ such that the number of indices $i$ for which $\mathsf{Encode}(\mathsf{ek}_i, \pi_i) \neq \mathsf{Decode}(\mathsf{td}_i, h)$ is more than $k^\epsilon$ for a universal $\epsilon < 1$. In other words, we allow for binding error in a small number of positions.

**HBG from VDTH.** A hidden bits generator consists of a setup phase that outputs a common reference string $\mathsf{crs}$, a hidden bit generation phase that takes in the $\mathsf{crs}$ and outputs a commitment $\mathsf{com}$, $k$ bits $e_1, \ldots, e_k$ along with proofs $\pi_1, \ldots, \pi_k$, and a verification phase that takes in $\mathsf{crs}$, $\mathsf{com}$, $(e_i, \pi_i)$ and either accepts or rejects the proof. We require this to satisfy two properties: hiding and binding. The hiding property requires that $e_{i^*}$ is pseudorandom even given $\mathsf{com}$ and $\{\pi_j\}_{j \neq i^*}$. The binding property requires that each $\mathsf{crs}$ is associated with a set $\mathcal{V}^{\mathsf{crs}}$ of $k$ bit strings such that any prover can only open to one of the strings in this set by giving valid proofs. Crucially, we require the size of this set to at most $2^{k^\nu}$ for some universal constant $\nu < 1$.

The transformation from VTDH to HBG is quite natural. The CRS of the HBG will be composed by the VDTH hash key $\mathsf{hk}$ and encoding keys $\{\mathsf{ek}_i\}_{i \in [k]}$ for each of the indices. The prover samples a random block vector $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_k)$ and hashes it to obtain the digest $h$ and the $k$ local openings $\pi_1, \ldots, \pi_k$. The digest $h$ will be the HBG commitment, whereas the proofs will be the local openings. The $i$-th hidden bit is computed as $e_i = \mathsf{Encode}(\mathsf{ek}_i, \pi_i)$. Finally, given a proof $\pi_i$ the verifier checks if this corresponds to a local opening with respect to the hash value $h$ using $\mathsf{Verify}$ and checks if $e_i = \mathsf{Encode}(\mathsf{ek}_i, \pi_i)$.

The completeness and the hiding properties of HBG follow directly from the completeness and the hiding properties of VTDH. To prove binding, we need to show that for a randomly sampled CRS, the number of possible strings that a corrupted prover could open to is at most $2^{k^\nu}$ for a universal constant $\nu < 1$. Note that for a fixed CRS of the HBG and a hash digest $h$, the set of decoded bits $(d_1, \ldots, d_k)$ is fixed. The statistical binding property of VTDH states that the prover cannot open to $(e_1, \ldots, e_k)$ such that the number of positions $i \in [k]$ where $e_i \neq d_i$ is more than $k^\epsilon$. This means that the set of strings that can be opened by any prover is within the Hamming ball of radius $k^\epsilon$ centered at $(d_1, \ldots, d_k)$. We show that the number of possible strings within this ball is at most $2^{k^\mu}$ for some constant $1 > \mu > \epsilon$. Now, from the succinctness property of VTDH, the set of all possible hash digests is at most $2^{\mathsf{poly}(\lambda)}$. Thus, the set of all possible strings that a prover can open to is bounded by $2^{\mathsf{poly}(\lambda)} \cdot 2^{k^\mu} < 2^{k^\nu}$ for a universal constant $\nu < 1$.

## 2.2 Recasting [Wat24] as a Vector Trapdoor Hash

Let us first explain how the recent construction of Hidden Bit Generator (HBG) from LWE due to Waters [Wat24] can be adapted to give a vector trapdoor hash. For the sake of explaining the core idea, we start with a simplified construction which doesn't satisfy the binding property. We will later explain how to add this property.

The hashing key $\mathsf{hk}$ comprises of a random LWE matrices $\mathbf{A}, \mathbf{B}_1, \ldots, \mathbf{B}_k$ along with discrete Gaussian matrices $\mathbf{W}_1, \ldots, \mathbf{W}_k$ such that $\mathbf{B}_i\mathbf{W}_i = \mathbf{A}$ for each $i \in [k]$.[4] This can be sampled by first sampling a random LWE matrix $\mathbf{A}$, then sampling matrices $\mathbf{B}_i$ along with their trapdoor, and then using the trapdoor to sample $\mathbf{W}_i$ from the conditional discrete Gaussian distribution [GPV08]. Without going into the actual dimensions of these matrices, we would like to remark that the number of rows in $\mathbf{B}_i$ is equal to the security parameter $\lambda$ and the number of columns in $\mathbf{A}$ and each $\mathbf{W}_i$ is $k \cdot \mathsf{poly}(\lambda)$ times the number of rows. The $i$-th encoding key $\mathsf{ek}_i$ is a random LWE sample $\mathbf{s}_i^T\mathbf{B}_i + \mathbf{e}_i^T$ and the trapdoor $\mathsf{td}_i$ is the LWE secret $\mathbf{s}_i$. This completes the description of the setup phase.

The hashing algorithm on input a binary vector $\mathbf{x}$ sets the hash $h$ to be $\mathbf{A}\mathbf{x}$ and the $i$-th local opening $\pi_i$ to be $\mathbf{W}_i\mathbf{x}$. Note that $\mathbf{W}_i$ is sampled according to the discrete Gaussian distribution and hence, each entry of $\mathbf{W}_i$ is small with overwhelming probability. Therefore, $\pi_i = \mathbf{W}_i\mathbf{x}$ is "short" as $\mathbf{x}$ is binary. The verification of the local opening simply checks if $\pi_i$ is "short" and if $\mathbf{B}_i\pi_i = h$. The encode algorithm computes $\lceil \mathsf{ek}_i\pi_i \rfloor_2$. The decode algorithm takes in the hash $h$ and the trapdoor $\mathbf{s}_i$ and outputs $\lceil \mathbf{s}_i^T h \rfloor_2$.

The correctness of the verification procedure and the succinctness property follow directly from the construction. The proof of hiding given in [Wat24] relies on LWE with super polynomial modulus-to-noise ratio. Here, we present a different proof which only relies on polynomial modulus-to-noise ratio. But as we will see later, the proof of correctness in [Wat24] still requires a super polynomial modulus-to-noise ratio and we require new techniques to overcome this limitation.

Let us now explain the new proof of hiding. Consider the following modified procedure for sampling the hashing key $\mathsf{hk}$. In this modified procedure, we first sample random LWE matrix $\mathbf{B}_{i^*}$ without its trapdoor and a random discrete Gaussian matrix $\mathbf{W}_{i^*}$. We set $\mathbf{A} = \mathbf{B}_{i^*}\mathbf{W}_{i^*}$. For all $i \neq i^*$, we sample $\mathbf{B}_i$ along with the trapdoor. We then reverse sample $\mathbf{W}_i$ from the discrete Gaussian distribution conditioned on $\mathbf{B}_i\mathbf{W}_i = \mathbf{A}$ using the trapdoor for $\mathbf{B}_i$. The output of the modified sampling procedure is statistically close to the original output from standard properties of sampling from conditional discrete Gaussian distribution [GPV08]. This modified sampling procedure allows to replace $\mathsf{ek}_{i^*}$ with a randomly sampled vector under the LWE assumption. To complete the hiding argument, observe that we can view the hash $h$ and $\{\pi_i\}_{i \neq i^*}$ as "leakage" on randomly chosen binary vector $\mathbf{x}$. If the length of $\mathbf{x}$ is large enough, then $\mathbf{x}$ has high min-entropy conditioned on this leakage. This follows since the number of columns in $\mathbf{A}$ and $\mathbf{W}_i$ is $k \cdot \mathsf{poly}(\lambda)$ factor more than the number of rows. Observe that $\mathsf{ek}_{i^*}\pi_{i^*} = \mathsf{ek}_{i^*}\mathbf{W}_{i^*}\mathbf{x}_{i^*} = \mathbf{u}_{i^*}^T\mathbf{x}_{i^*}$ where the distribution of $\mathbf{u}_{i^*}$ is statistically close to uniform.[5] We can now use leftover hash lemma to show that $\mathbf{u}_{i^*}^T\mathbf{x}_{i^*}$ is statistically close to random.

At a high-level, one would expect the binding property to follow from the fact that both $\mathbf{e}_i^T$ and $\pi_i$ are short. In a bit more detail, $\lceil \mathsf{ek}_i\pi_i \rfloor_2 = \lceil (\mathbf{s}_i^T\mathbf{B}_i + \mathbf{e}_i^T)\mathbf{W}_i\mathbf{x} \rfloor_2 = \lceil \mathbf{s}_i^T h + e_i^T\pi_i \rfloor_2 = \lceil \mathbf{s}_i^T h \rfloor_2$. However, this might not hold for an adversarially sampled $\pi_i$. To fix this, [Wat24] introduces an additional check as part of the verification procedure. Specifically, [Wat24] has an additional parameter called the $\mathsf{RoundingBound}$ and checks if $\lceil \mathsf{ek}_i\pi_i - \mathsf{RoundingBound} \rfloor_2 \stackrel{?}{=} \lceil \mathsf{ek}_i\pi_i \rfloor_2 \stackrel{?}{=} \lceil \mathsf{ek}_i\pi_i + \mathsf{RoundingBound} \rfloor_2$. This $\mathsf{RoundingBound}$ is chosen to be larger than the maximum absolute value that $\mathbf{e}_i^T\pi_i$ can take. If this check holds, then $\lceil \mathbf{s}_i^T h \rfloor_2 = \lceil \mathsf{ek}_i\pi_i - e_i^T\pi_i \rfloor_2 = \lceil \mathsf{ek}_i\pi_i \pm \mathsf{RoundingBound} \rfloor_2 =$

[4]We slightly modify the notation here from [Wat24] for consistency within our work.

[5]This can be proved using one more application of the conditional discrete Gaussian sampling property [GPV08].

$\lceil \mathsf{ek}_i \pi_i \rfloor_2$.

While the above fixes the issue with binding, it introduces an additional correctness error as honestly computed $\pi_i$ might not satisfy this additional rounding check. To mitigate this, [Wat24] introduces an additional randomly chosen element $d_i$ as part of the encoding key and sets the output of encode to be $\lceil \mathsf{ek}_i \pi_i + d_i \rfloor_2$ and that of decode to be $\lceil \mathbf{s}_i^T h + d_i \rfloor_2$. If the modulus $p$ is super polynomial factor more than the noise bound, it follows that a randomly chosen $d_i$ will make the rounding check pass with overwhelming probability.

**Drawbacks.** The above [Wat24] construction suffers from a couple of drawbacks.

– Firstly, it requires the LWE modulus to be a super-polynomial factor more than the noise bound for the correctness to hold.

– Secondly, the hashing key has a publicly testable structure. Specifically, one can test whether $\mathbf{B}_i \mathbf{W}_i = \mathbf{A}$ for each $i \in [k]$. Intuitively, we can think of this structure in terms of $k$-sided *star graph*. The center of the star is the matrix $\mathbf{A}$ and the $k$ other nodes are labeled with $\mathbf{B}_1, \ldots, \mathbf{B}_k$. The $k$ edges are labeled with $\mathbf{W}_1, \ldots, \mathbf{W}_k$ respectively. This structure intuitively captures the relation that $\mathbf{B}_i \mathbf{W}_i = \mathbf{A}$ for every $i \in [k]$.

  This star structure is quite crucial in the construction. Specifically, it allows us to connect the "short" hash given by $\mathbf{Ax}$ with the $k$ local openings given by $\mathbf{W}_1\mathbf{x}, \ldots, \mathbf{W}_k\mathbf{x}$. This connection is used in the proof of binding. It also allows us to replace a single $\mathsf{ek}_{i^*}$ with uniformly chosen random string by "programming" one edge and sampling the other edges conditioned on this edge.

**Relying on polynomial modulus-to-noise ratio.** We observe that the first drawback is relatively easy to fix. If the modulus is only a polynomial factor more than the noise bound, we get an inverse polynomial probability of error in the correctness. Importantly, we can set this inverse polynomial error parameter arbitrarily. This allows us to union bound over all indices $i \in [k]$ and still argue that the probability that there exists an index $i \in [k]$ such that the correctness error comes up in that index is inverse polynomial. Thus, we can sample several sets of $d_1, \ldots, d_k$. It follows that there exists at least one set with overwhelming probability such that the correctness error does not occur in that set. This set can be efficiently found and we can give the identity of this set as part of the hash. This allows us to obtain a construction whose security can be based on LWE with polynomial modulus-to-noise ratio.

However, overcoming the second drawback is more involved, and we discuss this next.

## 2.3 New Construction from Learning with Errors

We will now describe the ideas behind our new LWE-based instantiation.

**Hashing and encoding keys.** We give a new approach to remove the star structure in the hashing key. Specifically, our hashing key has the structure of $k$-matching. The end of points of the $i$-th edge in the matching are given by $\mathbf{B}_i, \mathbf{A}_i$ and the edge is labeled with $\mathbf{W}_i$. For each $i \in [k]$, we have $\mathbf{B}_i \mathbf{W}_i = \mathbf{A}_i$. However, without the star structure it is unclear how to even define the hash and the local openings while ensuring the succinctness property. Let us explain how we do it.

To create a hashing key $\mathsf{hk}$ we first sample uniform matrices

$$\mathbf{B}_1, \ldots, \mathbf{B}_k, \mathbf{A}_1, \ldots, \mathbf{A}_k$$

along with uniformly chosen binary matrices $\mathbf{W}_1, \ldots, \mathbf{W}_k$ such that $\mathbf{B}_i\mathbf{W}_i = \mathbf{A}_i$ for each $i \in [k]$. This can be efficiently done by first sampling $\mathbf{B}_i$ and $\mathbf{W}_i$ and setting $\mathbf{A}_i = \mathbf{B}_i\mathbf{W}_i$. Here, the number of rows in each $\mathbf{B}_i$ is the security parameter. Note that in the [Wat24] construction, it was crucial that $\mathbf{W}_i$ were sampled from the discrete Gaussian distribution. Here, we do not place such restrictions and looking ahead, we do not even employ lattice trapdoors in the construction or in the proof of security.

To create the $i$-th encoding key $\mathsf{ek}_i$ we first sample a LWE secret $\mathbf{s}_i$ and set $\mathbf{v}_i = (\mathbf{v}_{i,1}\|\ldots\|\mathbf{v}_{i,k})$ where

– $\mathbf{v}_{i,j}^T = \mathbf{s}_i^T\mathbf{B}_j + \mathbf{e}_{i,j}^T$ where $\mathbf{e}_{i,j}^T$ is a short error vector, for all $j \neq i$.

– $\mathbf{v}_{i,i}^T = \mathbf{s}_i^T\mathbf{A}_i + \mathbf{e}_{i,i}^T\mathbf{W}_i + \tilde{\mathbf{e}}_{i,i}^T$ where $\mathbf{e}_{i,i}^T$ and $\tilde{\mathbf{e}}_{i,i}^T$ are short error vectors.

The hash key, encoding keys and trapdoors are given by

$$\mathsf{hk} = \{\mathbf{B}_i, \mathbf{W}_i\}_{i\in[k]} \text{ and } \mathsf{ek}_i = \mathbf{v}_i \text{ and } \mathsf{td}_i = \mathbf{s}_i.$$

We first argue that the distribution of $(\mathsf{hk}, \{\mathsf{ek}_i\}_{i\in[n]})$ is pseudorandom under the LWE assumption, *without using any lattice trapdoors*. Note that it is enough to show that all $\mathbf{v}_i$ are indistinguishable from uniform vectors as $\mathbf{B}_i, \mathbf{W}_i$ are uniformly distributed. To show that $\mathbf{v}_i$ is pseudorandom, we first replace all $\mathbf{v}_{i,j}$ with uniform vectors, for $j \neq i$, and $\mathbf{v}_{i,i}$ by $\mathbf{u}^T\mathbf{W}_i + \tilde{\mathbf{e}}_{i,i}^T$ where $\mathbf{u}$ is uniformly chosen. This change goes unnoticed assuming that LWE is hard since

$$\begin{aligned}\left(\{\mathbf{s}_i^T\mathbf{B}_j + \mathbf{e}_{i,j}^T\}_{j\neq i}, \mathbf{s}_i^T\mathbf{A}_i + \mathbf{e}_{i,i}^T\mathbf{W}_i + \tilde{\mathbf{e}}_{i,i}^T\right) &= \left(\{\mathbf{s}_i^T\mathbf{B}_j + \mathbf{e}_{i,j}^T\}_{j\neq i}, (\mathbf{s}_i^T\mathbf{B}_i + \mathbf{e}_{i,i}^T)\mathbf{W}_i + \tilde{\mathbf{e}}_{i,i}^T\right)\\ &\approx_c \left(\{\mathbf{u}_j\}_{j\neq i}, \mathbf{u}^T\mathbf{W}_i + \tilde{\mathbf{e}}_{i,i}^T\right) \quad \text{(assuming LWE)}\end{aligned}$$

where $\{\mathbf{u}_j\}_{j\neq i}$ and $\mathbf{u}$ are uniformly chosen. And finally, we replace $\mathbf{u}^T\mathbf{W}_i + \tilde{\mathbf{e}}_{i,i}^T$ with uniform under the LWE assumption with binary LWE matrices [BLMR13].

**Hash, Encoding, and Decoding.** The hashing algorithm takes in $k$ binary vectors $\mathbf{x}_1, \ldots, \mathbf{x}_k$ and the hash $\mathbf{h}$ is defined as $\sum_i \mathbf{A}_i\mathbf{x}_i$. $\mathbf{h}$ is thus a vector with $\lambda$ rows, and thus the size of $\mathbf{h}$ is $\mathsf{poly}(\lambda)$ ensuring that the scheme is succinct.

The $i$-th local opening is given by

$$\pi_i = [\mathbf{W}_1\mathbf{x}_1\|\ldots\|\mathbf{W}_{i-1}\mathbf{x}_{i-1}\|\mathbf{x}_i\|\mathbf{W}_{i+1}\mathbf{x}_{i+1}\|\ldots\|\mathbf{W}_k\mathbf{x}_k].$$

In other words, we provide $\mathbf{x}_i$ in the clear in the $i$-th position and for every other position $j \in [k]$, we provide $\mathbf{W}_j\mathbf{x}_j$. The encoding and the decoding procedures are same as before. Namely, encoding is computed as $e_i = \lceil \mathbf{v}_i^T\pi_i \rfloor_2$ and decoding as $d_i = \lceil \mathbf{s}_i^T\mathbf{h} \rfloor_2$. Finally, the verification checks if (1) $\pi_i$ is "short"; and (2) $\mathbf{C}_i\pi_i \overset{?}{=} \mathbf{h}$, where $\mathbf{C}_i = [\mathbf{B}_1\|\ldots\|\mathbf{B}_{i-1}\|\mathbf{A}_i\|\mathbf{B}_{i+1}\|\ldots\|\mathbf{B}_k]$

**Hiding and statistical binding.** To argue that the scheme satisfies hiding at index $i^*$, we first switch the encoding key $\mathsf{ek}_{i^*}$ to a uniformly chosen string $\mathbf{v}_{i^*}$ arguing indistinguishability using the same argument as above, under the LWE assumption. Now, we view $\mathbf{W}_{i^*}\mathbf{x}_{i^*}$ as the leakage on $\mathbf{x}_{i^*}$.[6] Since $\mathbf{x}_{i^*}$ is uniformly chosen, it has high min-entropy conditioned on this leakage. We can now rely on leftover hash lemma to show that $\lceil \mathbf{v}_{i^*}^T \pi_{i^*} \rfloor_2$ is uniformly distributed from the fact that $\mathbf{v}_{i^*,i^*}^T \mathbf{x}_{i^*}$ is uniformly distributed.

Finally, we sketch how to prove statistical binding. Since we want the modulus $q$ to be polynomial, we will have to employ a different strategy than the one in [Wat24].

First, note that if verification passes then $\mathbf{C}_i\pi_i = \mathbf{h}$ and $\pi$ is short. This means that for all $i \in [k]$

$$e_i = \lceil \mathbf{v}_i^T \pi_i \rfloor_2 = \lceil (\mathbf{s}_i^T \mathbf{C}_i + \mathbf{f}_i^T)\pi_i \rfloor_2 = \lceil \mathbf{s}_i^T \mathbf{h} + \mathbf{e}_i^{*T} \rfloor_2$$

where $\mathbf{f}_i = (\mathbf{e}_{i,1} \| \dots \| \mathbf{e}_{i,i}\mathbf{W}_i + \tilde{\mathbf{e}}_{i,i} \| \dots \| \mathbf{e}_{i,k})$ and $\{\mathbf{e}_{i,j}\}_{j \in [k]}$ and $\tilde{\mathbf{e}}_{i,i}$ are short vectors of norm at most $B$. On the other hand $d_i = \lceil \mathbf{s}_i^T \mathbf{h} \rfloor_2$. We have that $e_i = d_i$ iff $\mathbf{s}_i^T\mathbf{h}$ is not in the set $\mathsf{Bad}_B$, which represents the set of elements close (i.e. at a distance $B$) of the rounding bound. When $\mathbf{s}_i^T\mathbf{h} \in \mathsf{Bad}_B$ we cannot guarantee $e_i = d_i$, i.e., there might be a binding error.

Second, recall that the definition of VTDH *already allows for a sublinear number of binding errors*. Hence, we just have to show that the probability that a hash value $\mathbf{h}$ exists for which there are more than $k^\epsilon$ binding errors is negligible over the random choice of $\mathsf{hk}, \{\mathsf{ek}_i\}_{i \in [k]}$, for some $\epsilon > 0$. To prove this, fix an hash value $h$. Then for an appropriate choice of parameters, there is a small probability (but still inverse polynomial) of $\mathbf{s}_i^T\mathbf{h} \in \mathsf{Bad}_B$.[7] Applying a Chernoff bound, we can make the probability of "having more than $k^\epsilon$ indices $i$ such that $\mathbf{s}_i^T\mathbf{h} \in \mathsf{Bad}$" to be negligible in $k$. Finally, we can union bound over all possible choices of $\mathbf{h}$. The resulting probability is negligible in $k$ as long as $k >> |\mathbf{h}|$. This analysis is similar to [Wat24, Appendix B].

## 2.4 Construction from DDH+LPN

In this subsection, we give our construction of vector trapdoor hash from DDH+LPN. We start with a natural adaptation of the LWE-based solution given above to the DDH setting and explain why it fails to satisfy hiding. We will later use LPN to fix the issue. However, this introduces several additional problems in proving binding and we finally explain how to overcome these issues.

**Natural Attempt.** Let us first consider the following adaptation of the LWE-based solution to the DDH setting. The hash key $\mathsf{hk}$ consists of $g^{\mathbf{B}_1}, \dots, g^{\mathbf{B}_k}$ where each $\mathbf{B}_i$ is randomly sampled from $\mathbb{Z}_p$ and randomly chosen compressing matrices $\mathbf{W}_1, \dots, \mathbf{W}_k$ in $\mathbb{Z}_p$. We define $\mathbf{A}_i = \mathbf{B}_i\mathbf{W}_i$. The $i$-th encoding key is given by $\mathsf{ek}_i = g^{\mathbf{s}_i^T\mathbf{C}_i}$ where $\mathbf{s}_i$ is randomly sampled and $\mathbf{C}_i = [\mathbf{B}_1 \| \dots \| \mathbf{B}_{i-1} \| \mathbf{A}_i \| \mathbf{B}_{i+1} \| \dots \| \mathbf{B}_k]$. The trapdoor $\mathsf{td}_i = \mathbf{s}_i$.

The hashing algorithm takes in $k$ binary vectors $\mathbf{x}_1, \dots, \mathbf{x}_k$ and the hash $h$ is defined as $g^{\sum_i \mathbf{A}_i\mathbf{x}_i}$.

[6]Note that given this leakage, the hash value $\sum_i \mathbf{A}_i\mathbf{x}_i$ can be efficiently computed.

[7]For simplicity, in this overview, we assume that the modulus $q$ is prime. Thus, we can use LHL to show that $\mathbf{s}_i^T\mathbf{h}$ is uniform over $\mathbb{Z}_q$, over the random choice of $\mathbf{s}_i$. In the main body, we show how to deal with arbitrary modulus.

The $i$-th local opening is given by

$$\pi_i = \begin{pmatrix} \mathbf{W}_1\mathbf{x}_1 \\ \vdots \\ \mathbf{W}_{i-1}\mathbf{x}_{i-1} \\ \mathbf{x}_i \\ \mathbf{W}_{i+1}\mathbf{x}_{i+1} \\ \vdots \\ \mathbf{W}_k\mathbf{x}_k \end{pmatrix}$$

The verification checks if $g^{\mathbf{C}_i\pi_i} \overset{?}{=} h$. The encode procedure outputs the first bit of $g^{s_i^T\mathbf{C}_i\pi_i}$. The decode procedure outputs the first bit of $g^{\mathbf{s}_i^T\sum_i \mathbf{A}_i\mathbf{x}_i}$.[8]

The correctness, succinctness, and the binding properties follow directly from the construction. Unfortunately, this construction does not satisfy hiding. Specifically, as $\mathbf{W}_i$ is compressing, the adversary can compute another matrix $\mathbf{V}_i$ such that $\mathbf{W}_i\mathbf{V}_i$ is the identity. It follows that $\mathbf{A}_i\mathbf{V}_i = \mathbf{B}_i$. The adversary can use this relation to compute $g^{\mathbf{s}_i^T[\mathbf{B}_1\|\ldots\|\mathbf{B}_k]}$ from $g^{\mathbf{s}_i^T\mathbf{C}_i}$. Given this, the $i$-th hidden bit is given by $g^{\mathbf{s}_i^T[\mathbf{B}_1\|\ldots\|\mathbf{B}_k]\cdot\mathbf{X}}$ where $\mathbf{X} = \begin{pmatrix} \mathbf{W}_1\mathbf{x}_1 \\ \vdots \\ \mathbf{W}_k\mathbf{x}_k \end{pmatrix}$. Note that the adversary can obtain $\mathbf{X}$ from the other local openings.

**LPN to the rescue.** We now slightly modify the construction and additionally rely on LPN over finite fields to make the hiding proof to go through. However, this will introduce additional issues in proving binding. We will later explain how to overcome these issues.

We make two modifications to the above construction. First, instead of setting the $i$-th encoding key $\mathsf{ek}_i$ as $g^{\mathbf{s}_i^T\mathbf{C}_i}$, we do the following. We first sample an LPN error vector $\mathbf{f}_i^T$ whose dimension is same as the number of columns in $\mathbf{A}_i$. We then set $\mathbf{e}_i^T$ to be a vector of same dimension as $\mathbf{s}_i^T\mathbf{C}_i$ such that it has $\mathbf{f}_i^T$ in the $i$-th slot and zeroes everywhere else. We set $\mathsf{ek}_i$ to be $g^{\mathbf{s}_i^T\mathbf{C}_i+\mathbf{e}_i^T}$. This implies that $\mathsf{ek}_i$ will be equal to $g^{[\mathbf{s}_i^T\mathbf{B}_1\|\ldots\|\mathbf{s}_i^T\mathbf{B}_{i-1}\|\mathbf{s}_i^T\mathbf{B}_i\mathbf{W}_i+\mathbf{f}_i^T\|\mathbf{s}_i^T\mathbf{B}_{i+1}\|\ldots\|\mathbf{s}_i^T\mathbf{B}_k]}$. Second, we sample $\mathbf{x}_1, \ldots, \mathbf{x}_k$ from the LPN error distribution instead of sampling them uniformly at random.

We now explain how we can show hiding with this modification. Note that the $i^*$-th encoding key $\mathsf{ek}_{i^*}$ can be generated as $g^{\mathbf{s}_{i^*}^T[\mathbf{B}_1\|\ldots\|\mathbf{B}_k]\cdot\mathbf{D}+\mathbf{e}_{i^*}^T}$ where

$$\mathbf{D} = \begin{pmatrix} \mathbf{I} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \ldots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \ldots & \mathbf{W}_{i^*} & \ldots \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \ldots & & \mathbf{I} \end{pmatrix}.$$

We first rely on DDH to replace $g^{\mathbf{s}_{i^*}^T[\mathbf{B}_1\|\ldots\|\mathbf{B}_k]}$ with $g^{[\mathbf{t}_1\|\ldots\|\mathbf{t}_k]}$ where $\mathbf{t}_i$ is uniformly sampled for each $i \in [k]$. We then rely on LPN over $\mathbb{Z}_p$ with secret $\mathbf{t}_{i^*}$, code matrix $\mathbf{W}_{i^*}$ and error vector $\mathbf{f}_{i^*}^T$ to

[8] If we were to output the first bit, then we need the property that the first bit is randomly distributed for a random group element. To remove this assumption, we use an extractor in the main body.

switch $\mathsf{ek}_{i^*}$ to $g^{[\mathbf{t}_1\|\cdots\|\mathbf{t}^*_{i^*}\|\cdots\|\mathbf{t}_k]}$ where $\mathbf{t}^*_{i^*}$ is randomly sampled. In other words, we used DDH and LPN to replace $\mathsf{ek}_{i^*}$ with a random vector of group elements. Now, consider the leakage $\mathbf{W}_{i^*}\mathbf{x}_{i^*}$ that is given to the adversary as part of the other local openings. Recall an earlier observation that this value is sufficient to efficiently compute the hash $h$. Since $\mathbf{W}_{i^*}$ is randomly sampled compressing matrix, we can rely on dual LPN to replace $\mathbf{W}_{i^*}\mathbf{x}_{i^*}$ with uniformly sampled $\mathbb{Z}_p$ elements. Hence, $\mathbf{x}_{i^*}$ has full min-entropy even conditioned on this leakage. Now, we can rely on leftover hash lemma to replace the output of the encode for $i^*$-th position with a uniformly chosen bit.

**Problem with Binding.** While LPN allowed us to argue the hiding property of the construction, it creates issues in arguing binding. Specifically, the output of the encode and the decode procedures for the $i$-th position are the same only if $\mathbf{f}_i^T\mathbf{x}_i = 0$. For a randomly sampled LPN error vectors $\mathbf{f}_i, \mathbf{x}_i$ this holds with inverse polynomial probability [Ale03]. However, note that $\mathbf{x}_i$ is sampled by an unbounded adversary. This adversary could break the LPN assumption, learn $\mathbf{f}_i$ in the clear, and sample a bad $\mathbf{x}_i$ such that $\mathbf{f}_i^T\mathbf{x}_i \neq 0$. This completely breaks the binding property.

**Our Solution: Reverse Randomization and Resilient Functions.** To fix this issue, we apply the technique of reverse randomization [DN00]. Specifically, for each position $i$, we sample multiple encoding keys $\mathsf{ek}_{i,1}, \ldots, \mathsf{ek}_{i,m}$ (for some parameter $m$ that is fixed later) using independently sampled LPN secrets $\mathbf{s}_{i,1}, \ldots, s_{i,m}$ and error vectors $\mathbf{f}_{i,1}, \ldots, \mathbf{f}_{i,m}$. We force the adversary to use the same $\pi_i$ for each one of these repetitions. By standard Chernoff bound followed by an union bound over $\mathbf{x}_i$, we ensure that the adversary can only bias a small number of these repetitions. For a large number of the repetitions, we have $\mathbf{f}_{i,j}^T\mathbf{x}_i = 0$ with overwhelming probability. However, we are not quite done.

Recall that the output of the encoding and the decoding procedures needs to be a single bit. What we essentially have is a sequence of $m$ bits $e_1, \ldots, e_m$ and $d_1, \ldots, d_m$ as the output of the encoding and decoding procedures respectively such that there is a small sized set $S \subset [m]$ where for every $j \notin S$, $e_j = d_j$. We need a mechanism to derive a single bit out of this sequence such the output remains the same for both inputs $(e_1, \ldots, e_m)$ and $(d_1, \ldots, d_m)$. For this purpose, we make use of resilient functions [BL85, AL93]. At a high-level, resilient functions are Boolean functions such that for a randomly sampled input, a small set of toggled positions does not change the output except with some small probability.

The work of Ajtai-Linial [AL93] gave a resilient function such that the probability that a set of input bits of size $s$ that is under adversarial control can change the value of the output on a randomly sampled input is at most $(\frac{\log^2 m}{m})s$. From the reverse randomization procedure, we can ensure that this $s$ is at most $m^\epsilon$ for a constant $\epsilon < 1$. This ensures that the probability that the adversary is able to bias a single position $i \in [k]$ is inverse polynomial.[9] It now follows from standard Chernoff bound that the adversary cannot bias more than $k^\epsilon$ positions among the set $[k]$ except with negligible probability. This would, in principle, complete the binding argument. Unfortunately, the resilient function of Ajtai-Linial is not explicit.

Recent progress due to Meka [Mek17] and Chattopadhyay-Zuckerman [CZ16] have provided explicit constructions that nearly match the non-explicit construction due to Ajtai-Linial. One drawback of these explicit constructions is that they are only "almost balanced". Specifically, the probability that the output of the function is 0 on a randomly chosen input is $1/2 \pm 1/10$. However, to

[9]Note that to make use of resilient functions, we need to ensure that the input to the function is randomly chosen. To argue that $d_1, \ldots, d_m$ are random for a fixed hash digest $h$, we rely on the fact that the LPN secrets are randomly chosen.

ensure that the hiding property holds, we need the output to be negligibly close to 1/2. To mitigate this, we take multiple independent copies of the output of the resilient function for each position and set the output of the encode to be the XOR of all these copies. By pile-up lemma, the output will be unbiased except with negligible probability. We carefully choose the number of copies such that the adversary still biases each position $i \in [k]$ with inverse polynomial probability. This allows us to use Chernoff bound to restrict the number of indices in $[k]$ that the adversary can toggle to be at most $k^{\epsilon}$ with overwhelming probability. This is sufficient to prove binding of VTDH.

We note that we have swept many details under the rug such as the choice of the number of repetitions for the reverse randomization, for the resilient function, etc. We refer the reader to the technical section for the details.

## 3 Preliminaries

Let $\lambda$ denote the cryptographic security parameter. We assume that all cryptographic algorithms implicitly take $1^{\lambda}$ as input. A function $\mu(\cdot) : \mathbb{N} \to \mathbb{R}^{+}$ is said to be negligible if for any polynomial $\mathsf{poly}(\cdot)$ there exists $\sec_0$ such that for all $\lambda > \lambda_0$ we have $\mu(\lambda) < \frac{1}{\mathsf{poly}(\lambda)}$. We will use $\mathsf{negl}(\cdot)$ to denote an unspecified negligible function and $\mathsf{poly}(\cdot)$ to denote an unspecified polynomial function.

Let $X$ and $Y$ be two distributions. Then i) $X \approx_c Y$ means that the distributions are computationally indistinguishable, ii) $X \approx_{\epsilon} Y$ means that their statistical distance is at most $\epsilon$ and, iii) $X \approx_s Y$ means that their statistical distance is at most $\mathsf{negl}(\lambda)$.

For a finite set $S$, we denote $x \leftarrow S$ as the process of sampling $x$ uniformly from the set $S$.

We state here the Chernoff bound that will be required in this work.

**Lemma 1 (Chernoff).** *Let $X_1 \ldots, X_n$ be i.i.d. random variables with mean at most $\varepsilon$, then*

$$\Pr\left[\sum X_i > 2\varepsilon n\right] < e^{-\varepsilon n/3}.$$

**Information theory.** The following definition and lemmas are taken from [DORS08].

**Definition 1 (Average Min-Entropy).** *For two jointly distributed random variables (X,Y), the average min-entropy of X conditioned on Y is defined as*

$$\tilde{H}_{\infty}(X|Y) = -\log(\mathbb{E}_{y \leftarrow \$ Y}[max_x \Pr[X = x|Y = y]]).$$

**Lemma 2 (Average min-entropy chain rule).** *For random variables $X, Y, Z$ where $Y$ is supported over a set of size $T$, we have*

$$\tilde{H}_{\infty}(X|(Y,Z)) \geq \tilde{H}_{\infty}((X,Y)|Z) - \log(T) \geq \tilde{H}_{\infty}(X|Z) - log(T).$$

**Lemma 3 (Leftover hashing).** *Let $X$ be a random variable supported on a finite set $\mathcal{X}$ such that $\tilde{H}_{\infty}(X) \geq k$. Let $h : \mathcal{S} \times \mathcal{X} \to \{0,1\}^{\ell}$ be a universal function, where $\ell \leq k - 2\log(1/\epsilon)$. Then for a uniformly chosen $S \leftarrow \mathcal{S}$*

$$(h(X), S) \approx_{\epsilon} (z, S)$$

*where $z \leftarrow \{0,1\}^{\ell}$.*

We will use the leftover hash lemma (LHL) over domains $\mathbb{Z}_q$ where $q$ is not necessarily prime. The following Lemma (due to Regev [Reg05]) allows us to use the LHL if the input domain is binary.

**Lemma 4 ([Reg05]).** *Fix a positive integer $q$ and let $\mathbf{A} \leftarrow_{\$} \mathbb{Z}_q^{n \times m}$ be chosen uniformly at random. Then the function $h : \mathbb{Z}_q^{n \times m} \times \{0,1\}^m \to \mathbb{Z}_q^n$ given by $h(\mathbf{A}, \mathbf{x}) = \mathbf{A}\mathbf{x}$ is a universal hash function.*

**Metric Spaces and Discrete Gaussians.** Let $\rho_s(\mathbf{x})$ be probability distribution of the Gaussian distribution over $\mathbb{R}^n$ with parameter $s$ and centered in $0$.

We define the discrete Gaussian distribution $D_{S,s}$ over $S$ and with parameter $s$ by the probability distribution $\rho_s(\mathbf{x})/\rho(S)$ for all $\mathbf{x} \in S$ (where $\rho_s(S) = \sum_{\mathbf{x}\in S} \rho_s(\mathbf{x})$).

We will use the following simple tail-bound for discrete gaussians, which follows from the fact that centered discrete gaussians are 0-subgaussian (see e.g. [MP12] Section 2.4).

**Lemma 5.** *Let $\mathbf{x} \leftarrow_{\$} D_{\mathbb{Z}^n,s}$. Then it holds that $\|\mathbf{x}\| < s \cdot \sqrt{\lambda n}$, except with probability $n \cdot e^{-\pi\lambda}$.*

We will also use the basic Cauchy-Schwartz inequality, which lets us bound the magnitude of an inner product by the products of the norms of its arguments.

**Lemma 6 (Cauchy-Schwartz Inequality).** *Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$. Then it holds that $|\langle \mathbf{x}, \mathbf{y}\rangle| \leq \|\mathbf{x}\| \cdot \|\mathbf{y}\|$.*

## 3.1 Resilient Functions

We define here the properties required for a Boolean function to be *balanced*, and *resilient*. We start by defining a balanced function.

**Definition 2 (Balanced Function).** *A Boolean function $F : \{0,1\}^\lambda \to \{0,1\}$ is said to be balanced if*

$$|\Pr_{x\leftarrow\{0,1\}^\lambda}[F(x) = 0] - 1/2| = \mathsf{negl}(\lambda).$$

We now define a resilient function. The influence of a set of input indices $Q$ is defined to be $I_Q(F)$ - denoting the probability that a Boolean function $F$'s output can be changed by controlling the inputs to the $Q$ indices. Specifically, we denote by $F_Q(x, y)$ the function output of $F$ when the bits indexed by $Q$ are set by $x \in \{0,1\}^{|Q|}$ and the others by $y \in \{0,1\}^{\lambda-|Q|}$. We say that $F$ is *determined* by a partial assignment $y \in \{0,1\}^{\lambda-|Q|}$ if there exists $b \in \{0,1\}$ such that $F_Q(x, y) = b$ for every $x$.

**Definition 3 (Resilient Function).** *For a Boolean function $F : \{0,1\}^\lambda \to \{0,1\}$, and $Q \subseteq [\lambda]$, let $I_Q(F)$ be the probability that $F$ is* not-determined *by a uniformly random partial assignment to the bits not in $Q$. Let $I_q(F) = \min_{Q\subseteq[\lambda],|Q|\leq q} I_Q(F)$. We say that $F$ is $(q, \delta)$-resilient if $I_q(F) \leq \delta$.*

We extend the above notion to a notion of a 'blockwise' resilient function. It extends the notion to boolean functions that work over $t$ 'blocks' of input. Here the function is only required to be resilient for a bounded influence for *each block*. We define this notion formally below.

**Definition 4 (Blockwise Resilient Function).** *For a boolean function $F : \{0,1\}^{\lambda\cdot t} \to \{0,1\}$, let $Q = (Q_1, \cdots, Q_t)$ where $Q_i \subseteq [\lambda]$ specifies the locations in the $i$-th input block. The blockwise influence $BI_Q(F)$ is the probability that $F$ is not determined by a uniformly partial assignment to the bits not in $Q$. Further, $BI_q(F) = \min_{Q=(Q_1,\cdots,Q_t),\forall i\in[t],Q_i\subseteq[\lambda],|Q_i|\leq q} BI_Q(F)$. We say that $F$ is $(q, \delta)$-blockwise resilient if $BI_q(F) \leq \delta$.*

We show that using resilient functions as an underlying primitive, we can indeed construct blockwise resilient functions that are also balanced. Specifically, we prove the following theorem in the Appendix A.

**Theorem 1 (Balanced Blockwise Resilient).** *For some universal constant $c' \geq 1$ the following holds. There exists a function $F : \{0,1\}^{n\cdot t} \mapsto \{0,1\}$ which can be computed in time $t \cdot n^{c'}$ such that,*

– *$F$ is balanced:* $\Pr_{x \leftarrow \$ \{0,1\}^{n \cdot t}}[F(x) = 0] = 1/2 \pm 1/2^t$.

– *$BI_q(F) \leq tc'q(\log^2 n)/n$.*

Note that for our application, it will be sufficient that $1/2^t$ is negligible in some parameter $\lambda$.

## 3.2 Hardness Assumptions

### 3.2.1 Learning with Errors

We now present the learning with errors assumption [Reg05].

**Definition 5 (Learning with Errors).** *Let $n, q \in \mathbb{Z}$. For all $m = \mathsf{poly}(n)$, we say that the LWE assumption $\mathsf{LWE}_{n,q,\sigma}$ is true if the following distributions are computationally indistinguishable:*

$$\left\{ (\mathbf{A}, \mathbf{s}^T\mathbf{A} + \mathbf{e}^T) \;\middle|\; \mathbf{A} \leftarrow \$ \, \mathbb{Z}_q^{n \times m}, \mathbf{s} \leftarrow \$ \, \mathbb{Z}_q^n, \mathbf{e} \leftarrow \$ \, D_{\mathbb{Z},\sigma}^m \right\}$$
$$\left\{ (\mathbf{A}, \mathbf{u}) \;\middle|\; \mathbf{A} \leftarrow \$ \, \mathbb{Z}_q^{n \times m}, \mathbf{u} \leftarrow \$ \, \mathbb{Z}_q^m \right\}$$

*where $D_{\mathbb{Z},\sigma}^m$ is the discrete gaussian distribution with parameter $\sigma$.*

When we consider $\sigma = \zeta q \geq 2\sqrt{n}$, the LWE problem is at least as hard as solving the approximate shortest independent vector problem to within a factor of $\tilde{O}(n/\zeta)$ [Reg05].

We will make use of the gadget-matrix [MP12]. For a modulus $q$, let $\mathbf{g} = (1, 2, 2^2, \ldots, 2^{\lceil \log(q) \rceil}) \in \mathbb{Z}^{1 \times \lceil \log(q) \rceil}$ be the powers-of-two vector (as a row vector). For a dimension $n$, let $\mathbf{I}$ be the $n \times n$ identity matrix. The gadget matrix $\mathbf{G}$ is given by

$$\mathbf{G} = \mathbf{I} \otimes \mathbf{g} = \begin{bmatrix} \mathbf{g} & & & \\ & \mathbf{g} & & \\ & & \ddots & \\ & & & \mathbf{g} \end{bmatrix} \in \mathbb{Z}^{n \times n\lceil \log(q) \rceil}.$$

Define the $\mathbf{g}^{-1} : \mathbb{Z}_q \to \{0,1\}^{\lceil \log(q) \rceil}$ as the function that computes the binary decomposition of $\mathbb{Z}_q$ elements (as a column vector). Note that it holds $\mathbf{g} \cdot \mathbf{g}^{-1}(x) = x$ This induces a function $\mathbf{G}^{-1} : \mathbb{Z}_q^n \to \{0,1\}^{n\lceil \log(q) \rceil}$ given by

$$\mathbf{G}^{-1}(\mathbf{x}) = \begin{bmatrix} \mathbf{g}^{-1}(x_1) \\ \vdots \\ \mathbf{g}^{-1}(x_n) \end{bmatrix}.$$

Note that it holds that $\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{x}) = \mathbf{x}$ For a matrix $\mathbf{A} = (\mathbf{a}_1, \ldots, \mathbf{a}_m) \in \mathbb{Z}_q^{n \times m}$ we define $\mathbf{G}^{-1}(\mathbf{A})$ via

$$\mathbf{G}^{-1}(\mathbf{A}) = (\mathbf{G}^{-1}(\mathbf{a}_1), \ldots, \mathbf{G}^{-1}(\mathbf{a}_m)).$$

Note finally that it holds that $\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{A}) = \mathbf{A}$.

### 3.2.2 Decisional Diffie-Hellman

In the following, let $\mathsf{Gen}$ be a (prime-order) *group generator*, that is, $\mathsf{Gen}$ is an algorithm that takes as an input a security parameter $1^\lambda$ and outputs $(\mathbb{G}, p, g)$, where $\mathbb{G}$ is the description of a multiplicative cyclic group, $p$ is the order of the group which is always a prime number unless differently specified, and $g$ is a generator of the group. In the following we state the decisional version of the Diffie-Hellman (DDH) assumption.

**Definition 6 (Decisional Diffie-Hellman Assumption).** *Let $(\mathbb{G}, p, g) \leftarrow\$ \mathsf{Gen}(1^\lambda)$. We say that the DDH assumption is true (with respect to $\mathsf{Gen}$) if the following distributions are computationally indistinguishable:*

$$\left\{(\mathbb{G}, p, g), (g^a, g^b, g^{ab}) \;\middle|\; a, b \leftarrow\$ \mathbb{Z}_p\right\}$$
$$\left\{(\mathbb{G}, p, g), (g^a, g^b, g^c) \;\middle|\; a, b, c \leftarrow\$ \mathbb{Z}_p\right\}.$$

For any matrix $\mathbf{A} \in \mathbb{Z}_p^{n\times m}$, the matrix of group elements is defined $\mathbf{H} = g^{\mathbf{A}} \in \mathbb{G}^{n\times m}$, where the $(i,j)$-th element of $\mathbf{H}$ is $g^{\mathbf{A}_{i,j}}$ where $\mathbf{A}_{i,j}$ is the corresponding $(i,j)$-th element of $\mathbf{A}$ in $\mathbb{Z}_p$. Further, for any matrix of group elements $\mathbf{H} = g^{\mathbf{A}} \in \mathbb{G}^{n\times m}$ and any matrix $\mathbf{B} \in \mathbb{Z}_p^{m\times \ell}$, $\mathbf{H}^{\mathbf{B}} = g^{\mathbf{AB}} \in \mathbb{G}^{n\times \ell}$.

We will additionally find it convenient to use the matrix DDH assumption assumption, which is implied by the DDH assumption [BHHO08].

**Definition 7 (Matrix Decisional Diffie-Hellman Assumption).** *Let $(\mathbb{G}, p, g) \leftarrow\$ \mathsf{Gen}(1^\lambda)$. We say that the matrix DDH assumption is true (with respect to $\mathsf{Gen}$) if the following distributions are computationally indistinguishable:*

$$\left\{(\mathbb{G}, p, g), (g^{\mathbf{A}}, g^{\mathbf{Au}}) \;\middle|\; \mathbf{A} \leftarrow\$ \mathbb{Z}_p^{n\times m}, \mathbf{u} \leftarrow\$ \mathbb{Z}_p^m\right\}$$
$$\left\{(\mathbb{G}, p, g), (g^{\mathbf{A}}, g^{\mathbf{v}}) \;\middle|\; \mathbf{A} \leftarrow\$ \mathbb{Z}_p^{n\times m}, \mathbf{v} \leftarrow\$ \mathbb{Z}_p^n\right\}.$$

### 3.2.3 Learning Parity with Noise

We start by defining the decisional version of the Learning Parity with Noise assumption over large fields. For a finite field $\mathbb{F}$, denote by $\mathsf{Ber}_\tau(\mathbb{F})$ the Bernoulli distribution which is obtained by sampling $0$ with probability $1-\tau$, and a uniformly random field element in $\mathbb{F}$ with probability $\tau$. For simplicity we write the definition in terms of $\mathbb{Z}_p$ for some prime $p$, where $\mathsf{Ber}_\tau(\mathbb{Z}_p)$ outputs a uniformly random non-zero element in $\mathbb{Z}_p$ with probability $\tau$.

**Definition 8 (Learning Parity with Noise over $\mathbb{Z}_p$).** *We say that $\mathsf{LPN}_{n,m,\tau}$ over $\mathbb{Z}_p$ is true if the following holds: For any constant $\ell > 0$ such $n = n(\lambda)$, $m = m(\lambda)$, $\tau = \tau(\lambda)$, a prime number $p = p(\lambda)$ of $\lambda^\ell$ bits it holds that the following distributions are computationally indistinguishable,*

$$\left\{(\mathbf{A}, u = \mathbf{sA} + \mathbf{e}) \;\middle|\; \mathbf{A} \leftarrow \mathbb{Z}_p^{n\times m}, \mathbf{s}^\top \leftarrow \mathbb{Z}_p^n, \mathbf{e} \leftarrow \mathsf{Ber}_\tau(\mathbb{Z}_p)^m\right\}_{\lambda\in[\mathbb{N}]}$$
$$\left\{(\mathbf{A}, u) \;\middle|\; \mathbf{A} \leftarrow \mathbb{Z}_p^{n\times m}, \mathbf{u} \leftarrow \mathbb{Z}_p^m\right\}_{\lambda\in[\mathbb{N}]}$$

**Dual LPN.** The corresponding *dual* version, which is equivalent to the above $\mathsf{LPN}_{n,m,\tau}$ assumption states that the distribution $\mathbf{eB}$ is computationally indistinguishable from a random vector in $\mathbb{Z}_p^{m-n}$, where $\mathbf{B} \in \mathbb{Z}_p^{m\times(m-n)}$ is the *parity-check matrix* of the matrix $\mathbf{A} \in \mathbb{Z}_p^{n\times m}$, i.e. $\mathbf{AB} = \mathbf{0}$. In the security game, a random matrix $\mathbf{A}$ is sampled, and a random parity-check matrix $\mathbf{B}$ for the corresponding $\mathbf{A}$ is sampled. The adversary is given $\mathbf{B}$ along with either $\mathbf{eB}$ or a random $\mathbf{v}$.

## 3.3 NIZKs in Hiddent Bit Model

We recall the definition of NIZKs in the hidden bit model.

**Definition 9.** *Let $\mathcal{L}$ be an* NP *language and $n$ be an integer. A Non-Interactive Zero-Knowledge Proof in the Hidden-Bits Model for $\mathcal{L}$ is given by a pair of PPT algorithms $(P, V)$, and a polynomial $k(\lambda, n)$, where:*

- *$P(1^\lambda, r, x, w)$: On input string $r \in \{0,1\}^{k(\lambda,n)}$, a statement $x$ of size $|x| = n$ and a witness $w$, output a set of indices $I \subseteq [k]$ and proof $\pi$.*
- *$V(1^\lambda, I, r_I, x, \pi)$: On input a subset $I \subseteq [k]$, a string $r_I$, a statement $x$ and a proof $\pi$, outputs* accept *or* reject*,*

*such that they satisfy the following properties:*

- ***Completeness:** We require that for all $x \in \mathcal{L}$ of size $|x| = n$ with witness $w$ we have:*

$$Pr[V(1^\lambda, I, r_I, x, \pi) = \mathsf{accept} : r \leftarrow \{0,1\}^{k(\lambda,n)}; (I, \pi) \leftarrow P(1^\lambda, r, x, w)] = 1;$$

- ***Soundness:** We require that for all polynomial $n = n(\lambda)$, and all unbounded cheating provers $\widetilde{P}$, we have:*

$$Pr[V(1^\lambda, I, r_I, x, \pi) = \mathsf{accept} \wedge x \notin \mathcal{L} \wedge |x| = n : r \leftarrow \{0,1\}^{k(\lambda,n)}; (x, \pi, I) \leftarrow \widetilde{P}(1^\lambda, r)] \leq \mathsf{negl}(\lambda)$$

- ***Zero-Knowledge:** We require that there exists an efficient simulator* Sim *such that for any adversary $\mathcal{A}$.the two following distributions are statistically indistinguishable:*

$$(I, r_I, \pi) \approx_s (I', r_{I'}, \pi')$$

*where $r \leftarrow \{0,1\}^{k(\lambda,n)}$, $(I, \pi) \leftarrow P(1^\lambda, r, x, w)$, $(I', r_{I'}, \pi') \leftarrow \mathsf{Sim}(1^\lambda, x)$.*

The following theorem was proved in [FLS90].

**Theorem 2 ([FLS90]).** *For some polynomial $k'(\lambda, n)$ and for any polynomial $q(\lambda, n)$, there exists NIZK in the hidden bit model with $k(\lambda, n) = k'(\lambda, n) \cdot q(\lambda, n)$ and soundness error $2^{-q(\lambda,n)} \cdot \mathsf{negl}(\lambda)$.*

## 3.4 Hidden-Bits Generator

The following definition of Hidden-Bits Generator (HBG) [QRW19] is taken from [Wat24]. For a string $\mathbf{r} \in \{0,1\}^k$ and a set of indices $I \subseteq [k]$, we denote by $\mathbf{r}_I$ the substring corresponding to the bits of $\mathbf{r}$ indexed by $I$.

**Definition 10 (Hidden-Bits Generator).** *A Hidden-Bits Generator (*$\mathsf{HBG}$*) is given by a set of* $\mathsf{PPT}$ *algorithms* $(\mathsf{GenSetup}, \mathsf{GenBits}, \mathsf{GenVerify})$*:*

- $\mathsf{GenSetup}(1^\lambda, 1^k)$ *outputs a common reference string* $\mathsf{crs}$.
- $\mathsf{GenBits}(\mathsf{crs})$ *outputs a triple* $(\mathsf{com}, \mathbf{r}, \{\pi_i\}_{i\in[k]})$ *where* $r \in \{0,1\}^k$.
- $\mathsf{GenVerify}(\mathsf{crs}, \mathsf{com}, i, \mathbf{r}_i, \pi_i)$ *outputs 1 or 0, where* $i \in [k]$.

*We require any Hidden-Bits Generator to satisfy the following properties.*

**Completeness.** *We require that for every polynomial* $k = k(\lambda)$ *and for all* $i \in [k]$*, we have:*

$$\Pr\left[\ \mathsf{GenVerify}(\mathsf{crs}, \mathsf{com}, i, \mathbf{r}_i, \pi) = 1 \ : \ \begin{array}{l}\mathsf{crs} \leftarrow \mathsf{GenSetup}(1^\lambda, 1^k)\\ (\mathsf{com}, \mathbf{r}, \{\pi_i\}_{i\in[k]}) \leftarrow \mathsf{GenBits}(\mathsf{crs})\end{array}\ \right] = 1.$$

**Binding:** *For every* $\mathsf{crs}$ *in the support of the algorithm* $\mathsf{GenSetup}(1^\lambda, 1^m)$*, there exists a set* $\mathcal{V}^{\mathsf{crs}}$ *with the following properties.*

**Output sparsity.** *There exists a universal constant* $\gamma < 1$ *and a fixed polynomial* $p(\cdot)$ *such that for every polynomial* $k = k(\lambda)$*, and every* $\mathsf{crs}$ *in the support of* $\mathsf{GenSetup}(1^\lambda, 1^k)$*,*

$$|\mathcal{V}^{\mathsf{crs}}| \le 2^{k^\gamma \cdot p(\lambda)}$$

**Statistical binding.** *For every (possibly inefficient) stateful adversary* $\mathcal{A}$

$$\Pr\left[\begin{array}{l}\mathbf{r}_I \notin \mathcal{V}_I^{\mathsf{crs}}\\ \forall i \in I,\ \mathsf{GenVerify}(\mathsf{crs}, \mathsf{com}, i, \mathbf{r}_i, \pi_i) = 1\end{array} \ : \ \begin{array}{l}1^k \leftarrow \mathcal{A}(1^\lambda)\\ \mathsf{crs} \leftarrow \mathsf{GenSetup}(1^\lambda, 1^k)\\ (\mathsf{com}, I, \mathbf{r}_I, \{\pi_i\}_{i\in I}) \leftarrow \mathcal{A}(\mathsf{crs})\end{array}\right] = \mathsf{negl}(\lambda).$$

*By* $\mathcal{V}_I^{\mathsf{crs}}$ *we denote the set* $\{\mathbf{r}_I : \mathbf{r} \in \mathcal{V}^{\mathsf{crs}}\}$*. We say that hidden-bits generator satisfies* computational binding *if the above is true only for computationally bounded adversaries.*

**Hiding.** *There is an alternate setup algorithm* $\mathsf{GenSetup}^*$ *that satisfies the following properties:*

**CRS Indistinguishability.** *We have*

$$\{\mathsf{crs} : \mathsf{crs} \leftarrow \mathsf{GenSetup}(1^\lambda, 1^k)\} \approx_c \{\mathsf{crs} : \mathsf{crs} \leftarrow \mathsf{GenSetup}^*(1^\lambda, 1^k)\}$$

**Pseudorandomness.** *We have that for all efficient and stateful adversaries* $\mathcal{A}$*,*

$$\left|\ \Pr[\mathcal{A}(\mathsf{crs}, \mathsf{com}, \{\mathbf{r}_i, \pi_i\}_{i\ne i^*}, \mathbf{r}_{i^*})] - \Pr[\mathcal{A}(\mathsf{crs}, \mathsf{com}, \{\mathbf{r}_i, \pi_i\}_{i\ne i^*}, u)]\ \right| = \mathsf{negl}(\lambda).$$

*where* $(1^k, i^*) \leftarrow \mathcal{A}(1^\lambda)$*,* $\mathsf{crs} \leftarrow \mathsf{GenSetup}^*(1^\lambda, 1^k)$*,* $(\mathsf{com}, \mathbf{r}, \{\pi_i\}_{i\in[k]}) \leftarrow \mathsf{GenBits}(\mathsf{crs})$ *and* $u \leftarrow \{0,1\}$*.*

**Definition 11.** *We say that hidden bits generator is dual-mode if the output of* $\mathsf{GenSetup}^*$ *is identically distributed to the uniform distribution and the pseudorandomness property holds against unbounded adversaries.*[10]

We give a sketch of the following theorem from [QRW19, KMY20] for the sake of completeness.

[10]We can also have the alternate version where we require the output of $\mathsf{GenSetup}$ to be identically distributed to the uniform distribution.

**Theorem 3 ([QRW19, KMY20]).** *Suppose there exists a (dual-mode) hidden-bits generator, then there exists a publicly verifiable, single theorem (dual-mode) NIZK.*

*Proof.* The construction is same as the one described in [QRW19] and uses a NIZK in the hidden bits model from Theorem 2. Let $k = k(\lambda, n) = k'(\lambda, n) \cdot q(\lambda, n)$ where we will set $q(\lambda, n)$ later. The CRS of the NIZK proof system comprises of the output of $\mathsf{GenSetup}(1^\lambda, 1^k)$ denoted by $\mathsf{crs}$ and a randomly sampled string $s \leftarrow \{0,1\}^k$. The prover first runs $\mathsf{GenBits}(\mathsf{crs})$ to obtain $\mathsf{com}, r, \{\pi_i\}_{i\in[k]}$. It then runs $P(1^\lambda, x, w, r \oplus s)$ to obtain $(I, \pi)$. The proof includes $(\mathsf{com}, I, r_I, \pi_I, \pi)$. The verifier runs $\mathsf{GenVerify}$ on $(\mathsf{crs}, \mathsf{com}, i, r_i, \pi_i)$ for each $i \in I$. If all the checks pass, it runs $V(1^\lambda, I, r_I \oplus s_I, x, \pi)$ and accepts only if $V$ accepts.

The completeness follows directly from the completeness of hidden bit generator and the completeness of the NIZK proof system in the hidden bit model.

We first prove soundness which follows nearly identically to the one described in [QRW19]. Recall that statistical binding property of the hidden bit generator guarantees the existence of a set $\mathcal{V}^{\mathsf{crs}}$ of size at most $2^{k^\nu \cdot \mathsf{poly}(\lambda)}$ such that the probability that any prover would be able to open to some string $r_I \notin \mathcal{V}_I^{\mathsf{crs}}$ is negligible. Let us condition on this bad event not happening. Let us call $r \in \{0,1\}^k$ to be $\mathsf{BAD}$ if there exists $(I, x, \pi)$ such that $x \notin \mathcal{L}$ and $V(1^\lambda, I, r_I, x, \pi)$ accepts. It follows from the soundness of NIZK in the hidden bit model that $\Pr[r \in \mathsf{BAD}] \leq 2^{-q(\lambda,n)} \cdot \mathsf{negl}(\lambda)$. Fix any $r^* \in \mathcal{V}^{\mathsf{crs}}$. The probability over the random choice of $s$ that $r \oplus s \in \mathsf{BAD}$ is at most $2^{-q(\lambda,n)} \cdot \mathsf{negl}(\lambda)$. By union bound over the set of all possible strings in $\mathcal{V}^{\mathsf{crs}}$, the probability that an adversary would be able to break soundness is at most $\mathsf{negl}(\lambda) + 2^{k^\nu \cdot \mathsf{poly}(\lambda)} 2^{-q(\lambda,n)} \cdot \mathsf{negl}(\lambda)$ where the first $\mathsf{negl}(\lambda)$ term is account for the adversary being able to open to some string not in $\mathcal{V}^{\mathsf{crs}}$. We set $q(\lambda, n) = (k'(\lambda, n)\mathsf{poly}(\lambda))^{\frac{1}{1-\nu}}$. Therefore,

$$k^\nu \cdot \mathsf{poly}(\lambda) = (k'(\lambda, n) q(\lambda, n))^\nu \cdot \mathsf{poly}(\lambda) \leq (q(\lambda, n))^\nu \cdot (q(\lambda, n))^{1-\nu}$$

Thus, soundness follows.

To argue hiding, we consider the following sequence of hybrids.

- $\mathsf{Hyb}_0$ : This corresponds to the execution of the proof system with the honest prover using the actual witness.

- $\mathsf{Hyb}_1$ : In this hybrid, we generate $\mathsf{crs}$ as the output of $\mathsf{GenSetup}^*$. This switch is indistinguishable from the CRS indistinguishability of hidden bit generators.

- $\mathsf{Hyb}_2$ : We make a syntactic change. We sample a random string $r^* \leftarrow \{0,1\}^k$ and run $P(1^\lambda, r^*, x, w)$. We compute $r$ as before and set $s = r \oplus r^*$. This hybrid is identical to the previous hybrid.

- $\mathsf{Hyb}_3$ : In this hybrid, we switch $r_{[k]\notin I}$ with uniformly chosen random string. Indistinguishability follows from pseudorandomness property of hidden bit generator as shown in [Wat24, Theorem 2.4]. The proof given in [Wat24] is for computationally bounded adversaries but if pseudorandomness holds against unbounded adversaries, then the exact same proof extends to showing that $\mathsf{Hyb}_2$ and $\mathsf{Hyb}_3$ are statistically close.

- $\mathsf{Hyb}_4$ : In this hybrid, we run the simulator $\mathsf{Sim}$ for the NIZKs in the hidden bit model to obtain $(I, r_I^*, \pi)$. We use this to generate the proof as in the previous hybrid. This hybrid is statistically close to the previous hybrid from the zero-knowledge property of NIZKs in the hidden bits model.

This completes the proof of hiding. Note that the above proof shows that if the hidden bit generator is dual-mode, then we get a single-theorem, dual-mode NIZK where the CRS is sampled as in $\mathsf{Hyb}_1$. $\square$

**Single-Theorem to Multi-Theorem in the Common Random String Model.** The standard FLS trick of using PRGs to transform single-theorem NIZKs to multi-theorem NIZKs does not preserve the dual-mode property. Specifically, if the output of $\mathsf{Setup}^*$ in the single-theorem case is identically distributed to the uniform distribution, then the same does not hold for the multi-theorem NIZK. Here, we give a different transformation that preserves this property under LWE.

First note that a single-theorem statistical NIZK is a statistically witness indistinguishable argument for any polynomial number of theorems, which follows by a simple hybrid argument. In order to boost a WI argument to a multi-theorem NIZK via the OR-proof methodology, we will additionally rely on a trapdoor language $\mathcal{L}_t \subseteq \{0,1\}^\ell$ which will be used by the simulator to simulate proofs. The trapdoor NP-language $\mathcal{L}_t$ will have the property that a uniformly chosen element from $\{0,1\}^\ell$ will be in $\mathcal{L}_t$ with all but negligible probability. More precisely, we require that there is an efficiently samplable distribution $\mathcal{Z}$ which outputs pairs $(z, td)$, such that $z$ is statistically close to uniform on $\{0,1\}^\ell$ and $td$ is a witness for $z \in \mathcal{L}_t$. Furthermore, we will choose $\mathcal{L}_t$ such that there is a pseudorandom distribution $\mathcal{X}$ on on its complement $\bar{\mathcal{L}_t}$

Equipped with such a distribution, the FLS transformation for a language $\mathcal{L}$ can be sketched as follows:

– The common reference string consists of a (uniformly chosen) CRS for the WI system as well as a uniformly chosen $z \leftarrow \{0,1\}^\ell$

– The simulator generates $z$ via $(z, td) \leftarrow_\$ \mathcal{Z}$ and keeps $td$ as a simulation trapdoor.

– Given a statement $x \in \mathcal{L}$ together with a witness $w$, the prover uses $w$ to compute a WI proof for the statement "$x \in \mathcal{L}$ or $z \in \mathcal{L}_t$"

– The simulator uses the simulation trapdoor $td$ to generate WI proofs for the statement "$x \in \mathcal{L}$ or $z \in \mathcal{L}_t$"

Clearly, since the underlying WI system is statistically witness-indistinguishable, the output distributions of a prover on input $(x, w)$ and the simulator on input $(x, td)$ are statistically close.

To establish computational soundness of this proof system, we sample $z$ from $\mathcal{X}$. As $\mathcal{X}$ is pseudorandom this modification is undetectable for a computationally bounded prover. Next, we switch the CRS of the underlying WI system to its statistically sound mode which also goes undetected by a computationally bounded prover. However as now $\mathcal{X}$ is supported on $\bar{\mathcal{L}_t}$, the statement $z \in \mathcal{L}_t$ is now false, and hence the statement "$x \in \mathcal{L}$ or $z \in \mathcal{L}_t$" implies "$x \in \mathcal{L}$". As the CRS of the underlying WI system is in statistically sound mode, the probability of a malicious prover providing a verifying proof for a false statement $x$ is thus negligible. Hence we have estabilished soundness.

It remains to provide a suitable trapdoor language $\mathcal{L}_t$ together with suitable distributions $\mathcal{Z}$ and $\mathcal{X}$. Let $\delta > 0$ be a distance parameter and $\mathbf{A} \in \mathbb{Z}_q^{n\times}$ be a matrix. The language $\mathcal{L}$ is given as follows:

$$\mathcal{L}_t = \{\mathbf{A} \in \mathbb{Z}_q^{n\times m} \mid \exists \mathbf{T} \in \mathbb{Z}^{m\times m} \text{ s.t. } \mathbf{T} \text{ has full rank, } \|\mathbf{T}\| \leq \delta \text{ and } \mathbf{A} \cdot \mathbf{T} = 0\}.$$

Our distribution $\mathcal{Z}$ will be given by a lattice trapdoor generator.

**Theorem 4 ([Ajt99, GPV08, MP12]).** *Fix a modulus $q$ and dimensions $n, m$ with $m \geq \Omega(n \log(q))$. There exists an efficient sampling algorithm* TrapGen*, which on input $q, n, m$ outputs matrices $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{T} \in \mathbb{Z}^{m \times m}$ such that*

- $\mathbf{A}$ *is distributed* $\mathsf{negl}(n)$ *close to uniform over* $\mathbb{Z}_q^{n \times m}$
- $\mathbf{T}$ *has full rank and it holds that* $\|\mathbf{T}\| \leq \delta = O(m)$
- *It holds that* $\mathbf{A} \cdot \mathbf{T} = 0$

First off, note that by the properties of the sampler TrapGen the distribution $\mathcal{Z}$ immediately satisfies the required properties, i.e. it is supported on $\mathcal{L}_t$ and statistically close to uniform. It remains to construct a pseudorandom distribution $\mathcal{X}$ which is supported on $\bar{\mathcal{L}}_t$ (except with negligible probability). We can easily construct this distribution via by relying on the LWE assumption. Let $q \geq \lambda\sigma\sqrt{m}\delta$.

- Choose $\mathbf{A}' \leftarrow_\$ \mathbb{Z}_q^{(n-1) \times m}$ uniformly at random.
- Choose $\mathbf{s} \leftarrow_\$ \mathbb{Z}_q^{n-1}$, $\mathbf{e} \leftarrow D_{\mathbb{Z}^m, \sigma}$.
- Compute and output
$$\mathbf{A} = \begin{pmatrix} \mathbf{A}' \\ \mathbf{s}\mathbf{A}' + \mathbf{e} \end{pmatrix}$$

Pseudorandomness of $\mathcal{X}$ follows immediately from the $\mathsf{LWE}(q, n, \sigma)$ assumption. Further note that matrices $\mathbf{A}$ generated by $\mathcal{X}$ are not supported on $\mathcal{L}_t$, except with negligible probability: Assume there exists a full-rank matrix $\mathbf{T}$ with $\|\mathbf{T}\| \leq \delta$ such that $\mathbf{A} \cdot \mathbf{T} = 0$. This means that

$$\mathbf{A}' \cdot \mathbf{T} = 0$$
$$\mathbf{s}\mathbf{A}'\mathbf{T} + \mathbf{e}\mathbf{T} = 0.$$

But these two equations imply that $\mathbf{e} \cdot \mathbf{T} = 0$. But since

$$\|\mathbf{e} \cdot \mathbf{T}\| \leq \|r\| \cdot \|\mathbf{T}\| \leq \|\mathbf{e}\| \cdot \delta,$$

the event that $\mathbf{e} \cdot \mathbf{T} = 0(\mod q)$ can only happen if $\|\mathbf{e}\| \geq q/\delta \geq \lambda\sigma\sqrt{m}$ or $\mathbf{e} = 0$. But both events have negligible probability. Hence the claim follows.

## 4 Vector Trapdoor Hash

In this section, we give the formal definition of vector trapdoor hash (VTDH) and show how to instanatiate the hidden bit generator (HBG) using this.

**Definition 12 (Vector Trapdoor Hash).** *A vector trapdoor hash (VTDH) is a tuple of algorithms* (Setup, Hash, Encode, Decode, Verify) *described as follows:*

- $\mathsf{Setup}(1^\lambda, 1^k)$ *takes as input a security parameter $\lambda$ and an integer $k \in \mathbb{N}$. It outputs a hash key* hk*, and $k$ pairs of encoding keys and trapdoors* $(\mathsf{ek}_1, \mathsf{td}_1), \ldots, (\mathsf{ek}_k, \mathsf{td}_k)$.

– $\mathsf{Hash}(\mathsf{hk}, \mathbf{x})$ *takes as input a hash key* hk *and an input* $\mathbf{x} \in \{0,1\}^{m(\lambda,k)}$*. It outputs a hash value* $h$ *and* $k$ *local openings* $\pi_1, \ldots, \pi_k$.

– $\mathsf{Encode}(\mathsf{ek}_i, \pi_i)$ *takes as input an encoding key* $\mathsf{ek}_i$ *and a local opening* $\pi_i$*. It outputs an encoding value* $e_i \in \{0,1\}$.

– $\mathsf{Decode}(\mathsf{td}_i, h)$ *takes as input a trapdoor* $\mathsf{td}_i$ *and a hash value* $h$*. It outputs a decoding value* $d_i \in \{0,1\}$.

– $\mathsf{Verify}(\mathsf{hk}, h, i, \pi_i)$ *takes as input a hash key* hk*, a hash value* $h$*, an index* $i \in [k]$ *and a local opening* $\pi_i$*. It outputs a bit* $b \in \{0,1\}$.

*We require VTDH to satisfy the following properties:*

– ***Completeness.*** *We require that for every polynomial* $k = k(\lambda)$ *and for all* $i \in [k]$*, we have:*

$$\Pr\left[ \mathsf{Verify}(\mathsf{hk}, h, i, \pi_i) = 1 \;:\; \begin{array}{l} \mathsf{hk}, \{(\mathsf{ek}_i, \mathsf{td}_i)\}_{i\in[k]} \leftarrow \mathsf{Setup}(1^\lambda, 1^k) \\ \mathbf{x} \leftarrow \{0,1\}^{m(\lambda,k)} \\ (h, \pi_1, \ldots, \pi_k) \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathbf{x}) \end{array} \right] = 1 - \mathsf{negl}(\lambda).$$

– ***Succinctness:*** *For all* $\lambda, k \in \mathbb{N}$ *and for every* $x \in \{0,1\}^{m(\lambda,k)}$*, the size of* $h$ *output by* $\mathsf{Hash}(\mathsf{hk}, x)$ *is at most* $\mathsf{poly}(\lambda)$.

– ***Statistical Binding.*** *There exists a universal constant* $0 < \epsilon < 1$ *such that for all* $\lambda, k \in \mathbb{N}$ *and for all unbounded adversaries* $\mathcal{A}$*, we have:*

$$\Pr\left[ |i \in [k] : e_i \neq d_i| > k^\epsilon \cdot \mathsf{poly}(\lambda) : \begin{array}{r} \mathsf{hk}, \{(\mathsf{ek}_i, \mathsf{td}_i)\}_{i\in[k]} \leftarrow \mathsf{Setup}(1^\lambda, 1^k) \\ (h, \{\pi_i\}_{i\in[k]}) \leftarrow \mathcal{A}(\mathsf{hk}, \{\mathsf{ek}_i\}_{i\in[k]}) \\ e_i \leftarrow \mathsf{Encode}(\mathsf{ek}_i, \pi_i) \quad \forall i \in [k] \\ \mathsf{Verify}(\mathsf{hk}, h, i, \pi_i) = 1 \quad \forall i \in [k] \\ d_i \leftarrow \mathsf{Decode}(\mathsf{td}_i, h) \quad \forall i \in [k] \end{array} \right] \leq \mathsf{negl}(\lambda).$$

– ***Hiding.*** *There is an alternate setup algorithm* $\mathsf{Setup}^*$ *that satisfies the following properties:*

**Mode Indistinguishability.** *We have*

$$\{(\mathsf{hk}, \{\mathsf{ek}_i\}_{i\in[k]}) : (\mathsf{hk}, \{\mathsf{ek}_i, \mathsf{td}_i\}_{i\in[k]}) \leftarrow \mathsf{Setup}(1^\lambda, 1^k)\} \approx_c$$
$$\{(\mathsf{hk}, \{\mathsf{ek}_i\}_{i\in[k]}) : (\mathsf{hk}, \{\mathsf{ek}_i, \mathsf{td}_i\}_{i\in[k]}) \leftarrow \mathsf{Setup}^*(1^\lambda, 1^k))\}.$$

**Pseudorandomness.** *For all* $\lambda \in \mathbb{N}$*, all* $k = \mathsf{poly}(\lambda)$*,* $i^* \in [k]$*, and all PPT adversaries* $\mathcal{A}$ *we have*

$$\left| \begin{array}{l} \Pr\left[ 1 \leftarrow \mathcal{A}(\mathsf{hk}, h, \{\mathsf{ek}_i, \pi_i\}_{i\neq i^*}, (\mathsf{ek}_{i^*}, r_{i^*})) : \begin{array}{r} \mathsf{hk}, \{(\mathsf{ek}_i, \mathsf{td}_i)\}_{i\in[k]} \leftarrow \mathsf{Setup}^*(1^\lambda, 1^k) \\ \mathbf{x} \leftarrow \{0,1\}^{m(\lambda,k)} \\ (h, \pi_1, \ldots, \pi_k) \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathbf{x}) \\ r_{i^*} \leftarrow \mathsf{Encode}(\mathsf{ek}_{i^*}, \pi_{i^*}) \end{array} \right] - \\ \\ \Pr\left[ 1 \leftarrow \mathcal{A}(\mathsf{hk}, h, \{\mathsf{ek}_i, \pi_i\}_{i\neq i^*}, (\mathsf{ek}_{i^*}, r_{i^*})) : \begin{array}{r} \mathsf{hk}, \{(\mathsf{ek}_i, \mathsf{td}_i)\}_{i\in[k]} \leftarrow \mathsf{Setup}^*(1^\lambda, 1^k) \\ \mathbf{x} \leftarrow \{0,1\}^{m(\lambda,k)} \\ (h, \pi_1, \ldots, \pi_k) \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathbf{x}) \\ r_{i^*} \leftarrow \{0,1\} \end{array} \right] \end{array} \right| \leq \mathsf{negl}(\lambda).$$

**Definition 13.** *We say that vector trapdoor hash is dual-mode if the output* $(\mathsf{hk}, \{\mathsf{ek}_i\}_{i\in[k]})$ *of* $\mathsf{Setup}^*$ *is identically distributed to the uniform distribution and the pseudorandomness property holds against unbounded adversaries.*

## 4.1 Vector Trapdoor Hash Imply Hidden Bits Generators

We show that a VTDH is sufficient to construct an HBG. We only state the theorem for the case of publicly verifiable VTDH, resulting in a publicly verifiable HBG. The designated-verifier case is identical, up to syntactical modifications.

**Theorem 5.** *Let* $(\mathsf{Setup}, \mathsf{Hash}, \mathsf{Encode}, \mathsf{Decode}, \mathsf{Verify})$ *be a (dual-mode) VTDH, then there exists a (dual-mode)* $\mathsf{HBG}$ *scheme* $(\mathsf{GenSetup}, \mathsf{GenBits}, \mathsf{GenVerify})$.

*Proof.* We describe the algorithms in the following.

– $\mathsf{GenSetup}(1^\lambda, 1^k)$: Run

$$(\mathsf{hk}, (\mathsf{ek}_1, \mathsf{td}_1), \ldots, (\mathsf{ek}_k, \mathsf{td}_k)) \leftarrow \mathsf{Setup}(1^\lambda, 1^k)$$

and define the crs to be $(\mathsf{hk}, \mathsf{ek}_1, \ldots, \mathsf{ek}_k)$.

– $\mathsf{GenBits}(\mathsf{crs})$: Sample $\mathbf{x} \leftarrow \{0,1\}^{m(\lambda,k)}$ uniformly and compute

$$(h, \pi_1, \ldots, \pi_k) \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathbf{x}) \text{ and } e_i \leftarrow \mathsf{Encode}(\mathsf{ek}_i, \pi_i) \quad \forall i \in [k].$$

Set $\mathsf{com} := h$ and $\mathbf{r}_i := e_i$.

– $\mathsf{Verify}(\mathsf{crs}, \mathsf{com}, i, \mathbf{r}_i, \pi_i)$: Return $1$ if

$$\mathsf{Verify}(\mathsf{hk}, h, i, \pi_i) \text{ and } \mathsf{Encode}(\mathsf{ek}_i, \pi_i) = \mathbf{r}_i$$

and $0$ otherwise.

Completeness is an immediate consequence of the completeness of the VTDH.

Similarly, crs indistinguishability and pseudorandomness follows from the corresponding properties of the VTDH, since for all indices $i^*$ it holds that:

$$(\mathsf{hk}, h, \{\mathsf{ek}_i, \pi_i\}_{i\neq i^*}, (\mathsf{ek}_{i^*}, \mathsf{Encode}(\mathsf{ek}_{i^*}, \pi_{i^*}))) \approx_c (\mathsf{hk}, h, \{\mathsf{ek}_i, \pi_i\}_{i\neq i^*}, (\mathsf{ek}_{i^*}, u_{i^*}))$$

where $u_{i^*} \leftarrow \{0,1\}$ is uniformly sampled.

Next, we prove that the HBG is binding. Let crs be $(\mathsf{hk}, \mathsf{ek}_1, \ldots, \mathsf{ek}_k)$ and the associated set of trapdoors be $\{\mathsf{td}_i\}_{i\in[k]}$. We define $\mathcal{V}^{\mathsf{crs}}$ as follows.

$$\mathcal{V}^{\mathsf{crs}} := \left\{ \mathbf{r} \in \{0,1\}^k : \exists\, h \text{ s.t. } \begin{array}{c} \mathbf{d}_i = \mathsf{Decode}(h, \mathsf{td}_i) \quad \forall i \in [k] \\ \mathsf{HW}(\mathbf{r} \oplus \mathbf{d}) \leq k^\epsilon \cdot \mathsf{poly}(\lambda) \end{array} \right\}.$$

where HW denotes the Hamming weight.

Let us first bound the size of $\mathcal{V}^{\mathsf{crs}}$. From succinctness property of VTDH, it follows that the $|h| = \mathsf{poly}(\lambda)$. Fixing the hash $h$ and the crs fixes the string $\mathbf{d}$. To bound the size of $\mathcal{V}^{\mathsf{crs}}$, we need to bound the number of strings that are Hamming distance at most $k^\epsilon \cdot \mathsf{poly}(\lambda)$ from the fixed string $\mathbf{d}$. This number is equal to:

$$\begin{aligned}\sum_{i=0}^{i=k^\epsilon \cdot \mathsf{poly}(\lambda)} \binom{k}{i} &\leq k^\epsilon \cdot \mathsf{poly}(\lambda) \binom{k}{k^\epsilon \cdot \mathsf{poly}(\lambda)} \\ &\leq k^\epsilon \cdot \mathsf{poly}(\lambda) (e \frac{k}{k^\epsilon \cdot \mathsf{poly}(\lambda)})^{k^\epsilon \cdot \mathsf{poly}(\lambda)} \\ &\leq k^\epsilon \cdot \mathsf{poly}(\lambda) 2^{k^\mu} \\ &\leq 2^{k^\nu}\end{aligned}$$

where $1 > \nu > \mu > \epsilon$. Thus, $|\mathcal{V}^{\mathsf{crs}}| \leq 2^{\mathsf{poly}(\lambda)} \cdot 2^{k^\nu}$ and output sparsity follows.

The statistical binding property immediately follows since for every $I \subseteq [k]$, $e_I$ that an adversary can open to using valid proofs $\pi_I$ belongs to $\mathcal{V}_I^{\mathsf{crs}}$. $\square$

**Remark 1.** *Analogously, if* (Setup, Hash, Encode, Decode, Verify) *be a DV-VTDH, then there exists an* DV-HBG *scheme* (GenSetup, GenBits, GenVerify). *The proof follows the same blueprint as the one presented above.*

## 5 Vector Trapdoor Hash from LWE

In this section, we give our construction of vector trapdoor hash from LWE.

**Parameters.** Let $q$ be a modulus, and let $m \geq n > 0$ and $k > 0$ be positive integers and let $\ell = n \cdot \log(q)$. Let $\mathbf{G} \in \mathbb{Z}_q^{n \times \ell}$ the gadget matrix of dimensions $n \times \ell$, and for a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ let $\mathbf{G}^{-1}(\mathbf{A}) \in \{0,1\}^{\ell \times m}$ be a binary matrix for which $\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{A}) = \mathbf{A}$ holds (c.f. Section 3.2.1).

**Construction.** We now describe our VTDH from the LWE assumption.
<u>$\mathsf{Setup}(1^\lambda, 1^k)$</u>

1. $\forall i \in [k]$, sample $\mathbf{B}_i \leftarrow \mathbb{Z}_q^{n \times \ell}$, $\mathbf{U}_i \leftarrow \mathbb{Z}_q^{n \times m}$, $\mathbf{W}_i = \mathbf{G}^{-1}(\mathbf{U}_i) \in \{0,1\}^{\ell \times m}$ and $\delta_i \leftarrow_\$ \mathbb{Z}_q$ uniformly random. Set $\mathbf{A}_i := \mathbf{B}_i \mathbf{W}_i \in \mathbb{Z}_q^{n \times m}$.
2. $\forall i \in [k]$
   (a) Sample $\mathbf{s}_i \leftarrow \mathbb{Z}_q^n$ and $\mathbf{e}_i \leftarrow D_{\mathbb{Z},\sigma}^{(k-1)\ell+m}$.
   (b) Set $\mathbf{C}_i := \mathbf{B}_1 \| \dots \| \mathbf{B}_{i-1} \| \mathbf{A}_i \| \mathbf{B}_{i+1} \| \dots \| \mathbf{B}_k \in \mathbb{Z}_p^{n \times ((k-1)\ell + m)}$.
   (c) For $j \in [k] \backslash \{i\}$ set $\mathbf{v}_{i,j} = \mathbf{s}_i \mathbf{B}_j + \mathbf{e}_{i,j}$ where $\mathbf{e}_i \leftarrow_\$ D_{\mathbb{Z},\sigma}^\ell$
   (d) Set $\mathbf{v}_{i,i} = \mathbf{s}_i \mathbf{A}_i + \mathbf{e}_{i,i} \mathbf{W}_i + \mathbf{e}'_{i,i}$ where $\mathbf{e}_{i,i} \leftarrow_\$ D_{\mathbb{Z},\sigma}^\ell$ and $\mathbf{e}'_{i,i} \leftarrow_\$ D_{\mathbb{Z},\sigma}^m$
   (e) Set $\mathbf{v}_i = (\mathbf{v}_{i,1} \| \dots \| \mathbf{v}_{i,k})$
   (f) Set $\mathsf{ek}_i = (\mathbf{v}_i, \delta_i)$ and $\mathsf{td}_i = (\mathbf{s}_i, \delta_i)$
3. Output $\mathsf{hk} = \{\mathbf{B}_i, \mathbf{W}_i, \mathsf{ek}_i\}_{i \in [k]}$ and $\{(\mathsf{ek}_i, \mathsf{td}_i)\}_{i \in [k]}$.

$\underline{\mathsf{Hash}(\mathsf{hk}, \mathbf{x} = \mathbf{x}_1 \| \ldots \| \mathbf{x}_k \in \mathbb{Z}_q^{m \times k})}$

1. Parse $\mathsf{hk} = \{\mathbf{B}_i, \mathbf{W}_i, \mathsf{ek}_i\}_{i \in [k]}$ and $\mathsf{ek}_i = \mathbf{v}_i$.
2. Set $\mathbf{A} := \mathbf{A}_1 \| \ldots \| \mathbf{A}_k$ where $\mathbf{A}_i = \mathbf{B}_i \mathbf{W}_i \quad \forall i \in [k]$
3. Set $\mathbf{h} := \mathbf{A}\mathbf{x} = \sum_{i=1}^{k} \mathbf{A}_i \mathbf{x}_i \in \mathbb{Z}_q^n$.
4. Set $\forall i \in [k]$, $\mathbf{y}_i = \mathbf{W}_i \mathbf{x}_i$ and $\mathbf{z}_i := \mathbf{y}_1 \| \ldots \| \mathbf{y}_{i-1} \| \mathbf{x}_i \| \mathbf{y}_{i+1} \| \ldots \| \mathbf{y}_k \in \mathbb{Z}_q^{(k-1)\ell + m}$.
5. Set $\pi_i = \mathbf{z}_i$
6. Output $(\mathbf{h}, \{\boldsymbol{\pi}_i\}_{i \in [k]})$.

$\underline{\mathsf{Encode}(\mathsf{ek}_i, \boldsymbol{\pi}_i)}$

1. Parse $\mathsf{ek}_i = \mathbf{v_i}$ and $\boldsymbol{\pi}_i(\mathbf{z}_i = \mathbf{y}_1 \| \ldots \| \mathbf{y}_{i-1} \| \mathbf{x}_i \| \mathbf{y}_{i+1} \| \ldots \| \mathbf{y}_k, \delta)$.
2. Compute $r_i := \lfloor \mathbf{v}_i^\top \mathbf{z}_i + \delta_i \rceil_2$.

$\underline{\mathsf{Decode}(\mathsf{td}_i, h)}$

1. Parse $\mathsf{td}_i = \mathbf{s}_i$ and $h = (\mathbf{h}, \delta)$.
2. Compute $d_i := \lfloor \mathbf{s}_i \mathbf{h} + \delta_i \rceil_2$.

$\underline{\mathsf{Verify}(\mathsf{hk}, \mathbf{h}, i, \boldsymbol{\pi}_i)}$

1. Parse $\boldsymbol{\pi}_i = \mathbf{y}_1 \| \ldots \| \mathbf{y}_{i-1} \| \mathbf{x}_i \| \mathbf{y}_{i+1} \| \ldots \| \mathbf{y}_k$.
2. Check if $\forall j \neq i \quad \mathbf{y_j} \in \{0, \ldots, m\}^\ell$ and $\mathbf{x} \in \{0,1\}^m$.
3. Check if $\mathbf{h} = \mathbf{C}_i \boldsymbol{\pi}_i$.
4. Output 1 if all the checks verify.

## 5.1 Completeness

We will first establish the completeness property of our scheme.

**Lemma 7 (Completeness).** *The scheme* (Setup, Hash, Encode, Decode, Verify) *provided above satisfies the completeness property.*

*Proof.* Clearly it holds that an honestly chosen $\mathbf{x}_i$ is in $\{0,1\}^m$. Moreover, as each $\mathbf{W}_j \in \{0,1\}^{\ell \times m}$ and $\mathbf{x}_j \in \{0,1\}^m$, it holds that $\mathbf{y}_j = \mathbf{W}_j \cdot \mathbf{x}_j \in \{0, \ldots, m\}^\ell$. Finally, it holds that

$$\begin{aligned} \mathbf{h} = \mathbf{A}\mathbf{x} &= \sum_j \mathbf{A}_j \mathbf{x}_j \\ &= \sum_{j \neq i} \mathbf{B}_j \mathbf{W}_j \mathbf{x}_j + \mathbf{A}_i \mathbf{x}_i \\ &= \mathbf{C}_i \pi_i, \end{aligned}$$

i.e. also the last check always passes. We conclude that the scheme satisfies the completeness property. □

## 5.2 Pseudorandomness of Hashing and Encoding Keys

We will now establish that he hashing and encoding keys are jointly pseudorandom.

**Lemma 8.** *Let $q$ be a modulus and assume that* $\mathsf{LWE}(q, n, \sigma)$ *holds. Then the tuple* $(\mathsf{hk}, \{\mathsf{ek}_i\}_{i\in[k]})$ *is pseudorandom.*

*Proof.* First note that the $\mathbf{A}_i$ are chosen uniformly random. We will hence establish that the $\mathsf{ek}_i$ are pseudorandom given the $\mathbf{A}_i$. We will establish the claim via a hybrid argument, successively replacing the $\mathsf{ek}_i = (\mathbf{v}_i, \delta_i)$ by uniformly random strings. As the $\mathsf{ek}_i$ are generated independently, it thus suffices to show that a single $\mathsf{ek}_{i^*}$ is pseudorandom given the $\mathbf{A}_i$. Consider the following (sub-)hybrids.

$\mathsf{Hyb}_0$ : This is the real experiment. In this experiment $\mathbf{v}_{i^*} = (\mathbf{v}_{i^*,1}, \ldots, \mathbf{v}_{i^*,k})$ is computed via $\mathbf{v}_{i^*,j} = \mathbf{s}_i\mathbf{B}_j + \mathbf{e}_{i^*,j}$ for $j \neq i^*$ and $\mathbf{v}_{i^*,i^*} = \mathbf{s}_i\mathbf{A}_i + \mathbf{e}_{i^*,i^*}\mathbf{W}_{i^*} + \mathbf{e}'_{i^*,i^*}$

$\mathsf{Hyb}_1$ : In this hybrid we modify the generation of the encoding key $\mathsf{ek}_{i^*} = ((\mathbf{v}_{i^*,1}\|\ldots\|\mathbf{v}_{i^*,k}), \delta_i)$ as follows. For all $j \neq i^*$ we choose $\mathbf{v}_{i^*,j}$ uniformly at random, and we compute $\mathbf{v}_{i^*,i^*}$ by chosing a uniformly random $\mathbf{u} \in \mathbb{Z}_q^\ell$ and setting $\mathbf{v}_{i^*,i^*} = \mathbf{u}\mathbf{W}_i + \mathbf{e}'_{i^*,i^*}$.

<u>$\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_0$.</u> Computational indistinguishability of hybrids 0 and 1 follows from the $\mathsf{LWE}(q, n, \sigma)$ assumption: In hybrid 0 we can write $\mathbf{v}_{i^*,i^*}$ as

$$\mathbf{v}_{i^*,i^*} = (\mathbf{s}_i\mathbf{B}_i + \mathbf{e}'_{i^*,i^*})\mathbf{W}_i + \mathbf{e}'_{i^*,i^*}.$$

Hence, under the $\mathsf{LWE}(q, n, \sigma)$ we can replace all terms $\mathbf{s}_i\mathbf{B}_i + \mathbf{e}'_{i^*,i^*}$ with uniformly random vectors in $\mathbb{Z}_q^\ell$ and the claim follows.

$\mathsf{Hyb}_2$ : This is identical to hybrid 1, except that we now choose $\mathbf{v}_{i^*,i^*}$ uniformly random instead of generating it via $\mathbf{v}_{i^*,i^*} = \mathbf{u}\mathbf{W}_i + \mathbf{e}_{i^*,i^*}$ for a uniformly random $\mathbf{u} \leftarrow_\$ \mathbb{Z}_q^\ell$.

<u>$\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_1$.</u> We claim that hybrids 1 and 2 are indistinguishable under the $\mathsf{LWE}(q, n, \sigma)$ assumption. Assume towards contradiction that $\mathcal{A}$ distinguishes between hybrids 1 and 2 with non-negligible advantage.

The reduction $\mathcal{R}$ proceeds as follows. Given an LWE challenge $(\mathbf{A}, \mathbf{z})$, where $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ is a uniformly random matrix and $\mathbf{z}$ is either of the form $\mathbf{s}\mathbf{A}+\mathbf{e}$ or chosen uniformly, the reduction proceeds as follows: It simulates hybrid 1 faithfully, except that it sets $\mathbf{W}_{i^*} = \mathbf{G}^{-1}(\mathbf{A})$ and sets $\mathbf{v}_{i^*,i^*} = \mathbf{z} + \mathbf{u}'\mathbf{W}_{i^*}$, where $\mathbf{u}' \leftarrow_\$ \mathbb{Z}_q^\ell$ is chosen uniformly random. It continues simulating hybrid 1 and outputs whatever $\mathcal{A}$ outputs. Note that since $\mathbf{A}$ is random matrix in $\mathbb{Z}_q^{n\times m}$, $\mathbf{W}_i^*$ has the same distribution as in $\mathsf{Hyb}_1$, i.e. we have just set $\mathbf{U}_{i^*} = \mathbf{A}$.

First assume that $\mathbf{z} = \mathbf{s}\mathbf{A} + \mathbf{e}$, i.e. $\mathbf{z}$ is an LWE sample. Then we can equivalently write $\mathbf{z}$ as $\mathbf{s}\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{A}) + \mathbf{e}$. That is, we can write $\mathbf{v}_{i^*,i^*}$ as

$$\begin{aligned}\mathbf{v}_{i^*,i^*} &= \mathbf{z} + \mathbf{u}'\mathbf{W}_{i^*}\\ &= \mathbf{s}\mathbf{G}\cdot\mathbf{W}_{i^*} + \mathbf{e} + \mathbf{u}'\mathbf{W}_{i^*}\\ &(\mathbf{s}\mathbf{G} + \mathbf{u}')\mathbf{W}_{i^*} + \mathbf{e}\\ &\equiv \mathbf{u}\mathbf{W}_{i^*} + \mathbf{e}\end{aligned}$$

where the second equality follows as $\mathbf{W}_{i^*} = \mathbf{G}^{-1}(\mathbf{A})$ and the last line follows as $\mathbf{s}\mathbf{G} + \mathbf{u}'$ is distributed uniformly random. We conclude that $\mathcal{R}$'s simulation is distributed identical to hybrid 1. On the other hand, if $\mathbf{z}$ is distributed uniformly random, then so is $\mathbf{v}_{i^*,i^*} = \mathbf{z}+\mathbf{u}'\mathbf{W}_{i^*}$ and we conclude that $\mathcal{R}$'s simulation is distributed identical to hybrid 2. Thus, $\mathcal{R}$ distinguishes $\mathsf{LWE}(q, n, \sigma)$ with non-negligible advantage and we arrive at the desired contradiction.

Thus, in $\mathsf{Hyb}_2$ the encoding key $\mathsf{ek}_{i^*}$ is distributed uniformly random, which concludes this proof. □

## 5.3 Statistical Hiding

We will now establish the hiding property of the vector trapdoor hash ($\mathsf{Setup}, \mathsf{Hash}, \mathsf{Encode}, \mathsf{Decode}, \mathsf{Verify}$).

**Lemma 9.** *Let $q = 2q'$ be an even modulus. Further let $\ell \leq (m - \log(q) - 2\lambda)/log(m)$. Assume that* $\mathsf{hk}$ *and the* $\{\mathsf{ek}_i\}_{i\in[k]}$ *are chosen uniformly random. Then* ($\mathsf{Setup}, \mathsf{Hash}, \mathsf{Encode}, \mathsf{Decode}, \mathsf{Verify}$) *satisfies the statistical hiding property.*

*Proof.* Fix an index $i^*$ and let $\mathcal{A}$ be a PPT distinguisher against the hiding experiment. Consider the following hybrids.

$\mathsf{Hyb}_0$ : This is the real experiment.

$\mathsf{Hyb}_1$ : This hybrid is identically distributed to hybrid 0, except that we choose $r_{i^*}$ uniformly random.

$\underline{\mathsf{Hyb}_1 \approx_s \mathsf{Hyb}_0}$. We claim that hybrid 1 is statistically close to hybrid 0. First observe that $\mathbf{x}_{i^*}$ has high conditional min-entropy given $\mathbf{W}_{i^*}$ and $\mathbf{y}_{i^*} = \mathbf{W}_{i^*}\mathbf{x}_{i^*}$. Specifically, fix any $\mathbf{W}_{i^*} \in \{0,1\}^{\ell\times m}$. As $\mathbf{W}_{i^*} \cdot \mathbf{x}_{i^*}$ is supported on $\{0,\dots,m\}^\ell$ (as both $\mathbf{W}_{i^*}$ and $\mathbf{x}_{i^*}$ are binary)
it holds by the min-entropy chain rule (Lemma 2) that

$$H_\infty(\mathbf{x}_{i^*}|\mathbf{W}_{i^*}\mathbf{x}_{i^*}) \geq H_\infty(\mathbf{x}_{i^*}) - \ell \cdot \log(m) = m - \ell \cdot \log(m) \geq \log(q) + 2\lambda..$$

Consequently, as $\mathbf{x}_{i^*}$ is binary the leftover hash lemma (Lemma 3 and Lemma 4) yields that

$$(\mathbf{v}_{i^*,i^*}, \mathbf{v}_{i^*,i^*} \cdot \mathbf{x}_{i^*}) \approx_{2^{-\lambda}} (\mathbf{v}_{i^*,i^*}, u),$$

for a uniformly random $u \leftarrow_\$ \mathbb{Z}_q$. It follows that the encoding $r_{i^*} = \lceil \mathbf{v}_{i^*,i^*} \cdot \mathbf{x}_{i^*} + \delta - i^* \rfloor$ is also statistically close to uniform as $q$ is even (and hence the most significant bit of $u$ is also uniform).

Finally, in the remaining hybrids 4 and 5 we undo the hybrid changes made in hybrids 1 and 2. Hybrid 5 is thus identical to the ideal experiment. □

## 5.4 Statistical Binding

We will now establish the statistical binding property for the vector trapdoor hash ($\mathsf{Setup}, \mathsf{Hash}, \mathsf{Encode}, \mathsf{Decode}, \mathsf{Verify}$)

**Lemma 10.** *Given that $q \geq \Omega(k^{2-\epsilon}\sigma m\ell\lambda^{1/2-c})$ and $n \leq O(k^\epsilon\lambda^c/\log(\lambda))$ for some constant $c > 0$ it holds that* ($\mathsf{Setup}, \mathsf{Hash}, \mathsf{Encode}, \mathsf{Decode}, \mathsf{Verify}$) *satisfies the statistical binding property.*

*Proof.* Note that it holds that

$$\begin{aligned}
\mathbf{v}_i \cdot \pi_i &= \sum_{j \neq i} \mathbf{v}_{i,j} \cdot \mathbf{W}_j \mathbf{x}_j + \mathbf{v}_{i,i} \cdot \mathbf{x}_i \\
&= \sum_{j \neq i} (\mathbf{s}_i \mathbf{B}_j + \mathbf{e}_{i,j}) \cdot \mathbf{W}_j \mathbf{x}_j + (\mathbf{s}_i \cdot \mathbf{B}_i \cdot \mathbf{W}_i + \mathbf{e}_{i,i} \mathbf{W}_i + \mathbf{e}'_{i,i}) \cdot \mathbf{x}_i \\
&= \mathbf{s}_i \sum_j \mathbf{B}_j \mathbf{W}_j \mathbf{x}_j + \sum_j \mathbf{e}_{i,j} \mathbf{W}_j \mathbf{x}_j + \mathbf{e}'_{i,i} \cdot \mathbf{x}_i \\
&= \mathbf{s}_i \cdot \sum_j \mathbf{A}_j \mathbf{x}_j + \sum_j \mathbf{e}_{i,j} \mathbf{W}_j \mathbf{x}_j + \mathbf{e}'_{i,i} \cdot \mathbf{x}_i \\
&= \mathbf{s}_i \mathbf{h} + \sum_j \mathbf{e}_{i,j} \mathbf{W}_j \mathbf{x}_j + \mathbf{e}'_{i,i} \cdot \mathbf{x}_i.
\end{aligned}$$

First note by Lemma 5, the $L_2$-norm of the concatenation $(\mathbf{e}_{i,1} \| \ldots \| \mathbf{e}_{i,k})$ is bounded by $\sqrt{\lambda \ell k} \cdot \sigma$, except with negligible probability $e^{-\lambda/2}$. Likewise, the norm of the $\mathbf{e}'_{i,i}$ is bounded by $\sqrt{\lambda m} \cdot \sigma$, except with probability $e^{-\lambda/2}$.

As $\mathsf{Verify}(\mathsf{hk}, \mathbf{h}, i, \boldsymbol{\pi}_i) = 1$, it holds that $(\mathbf{W}_1 \mathbf{x}_1 \| \ldots \| \mathbf{W}_k \mathbf{x}_k) \in \{0, \ldots, m\}^{\ell k}$ and $\mathbf{x}_i \in \{0,1\}^m$. Hence it holds that $\|(\mathbf{W}_1 \mathbf{x}_1 \| \ldots \| \mathbf{W}_k \mathbf{x}_k)\|_2 \leq \sqrt{\ell k} \cdot m$ and hence $\|\mathbf{x}_i\|_2 \leq \sqrt{m}$. We obtain via the Cauchy-Schwartz inequality (Lemma 6) and the triangle inequality that

$$B = \left| \sum_j \mathbf{e}_{i,j} \mathbf{W}_j \mathbf{x}_j + \mathbf{e}'_{i,i} \cdot \mathbf{x}_i \right| \leq \sqrt{\lambda} \sigma m (\ell k + 1) \leq 2 \sqrt{\lambda} \sigma m \ell k.$$

Now fix any hash-value $\mathbf{h}$ and let $\mathsf{hk}$ and $\{(\mathsf{ek}_i, \mathsf{td}_i)\}_{i \in [k]}$ be the output of the $\mathsf{Setup}(1^\lambda, 1^k)$ algorithm run on fresh random coins. Parse $\mathsf{td}_i = (\mathbf{s}_i, \delta)$.

For each index $i$, given that $\mathbf{s}_i \mathbf{h} + \delta_i \notin \pm q/4 + [-B, B]$, it holds that

$$\lceil \mathbf{v}_i \cdot \pi_i + \delta_i \rfloor = \lceil \mathbf{s}_i \mathbf{h} + \delta_i \rfloor.$$

Say that an index $i \in [k]$ is *bad* if $\mathbf{s}_i \mathbf{h} + \delta_i \in \pm q/4 + [-B, B]$. For each index $i$, the probability that it is bad is $4B/q$ as $\delta_i$ is chosen uniformly random. Thus, in expectation there are $4kB/q$ bad indices.

Note that the due to the independence of the $\delta_i$ the events that indices are bad are independent. Hence, a Chernoff bound yields that there are less than $8kB/q$ bad indices for this specific hash value $\mathbf{h}$, except with probability $e^{-4kB/(3q)}$.

The hash value $\mathbf{h}$ is in $\mathbb{Z}_q^n$, i.e. there are exactly $q^n$ choices for $\mathbf{h}$. Hence, by a union-bound the probability over the choice of honestly generated $\mathsf{hk}$ and $\{(\mathsf{ek}_i, \mathsf{td}_i)\}_{i \in [k]}$ that there exists a hash value $\mathbf{h}$ for which there are more than $4kB/q$ bad indices is at most

$$q^n \cdot e^{-4kB/(3q)} = e^{n \cdot \ln(q) - 4kB/(3q)}.$$

Recall that we need to bound the number of bad indices with $k^\epsilon$ (for $0 < \epsilon < 1$). Hence we have the following constraints on $q$:

$$8kB/q \leq k^\epsilon \lambda^c \tag{1}$$

$$4kB/(3q) - n \ln(q) \geq \omega(\log(\lambda)). \tag{2}$$

We can satisfy inequality (1) by choosing $q \geq 8k^{1-\epsilon}B/\lambda^c = \Omega(k^{2-\epsilon}\sigma m\ell\lambda^{1/2-c})$. Furthermore, we can satisfy inequality (2) by e.g. choosing

$$\begin{aligned} n &\leq kB/(q\log(q)) \\ &\leq O(k^{\epsilon}\lambda^{c}/\log(\lambda)) \end{aligned}$$

□

## 5.5 Setting the Parameters

1. By Lemma 9 we need to choose $\ell \leq (m - \log(q) - 2\lambda)/log(m)$, as well as $n \geq \lambda$ and $\ell \geq n\log(q)$ to get hardness for LWE$(q, n, \sigma)$.

2. By Lemma 10 we need to choose $q \geq \Omega(k^{2-\epsilon}\sigma m\ell\lambda^{1/2-c})$ and $n \leq O(k^{\epsilon}\lambda^{c}/\log(\lambda))$ for some constants $0 \leq \epsilon < 1$ and $c \geq 0$.

Let $k$ be a free parameter for now. We can choose $n = \lambda$, which satisfies $n \leq O(k^{\epsilon}\lambda^{c}/\log(\lambda))$ for e.g. $c = 2$. We choose $\sigma$ by $\sigma = \Omega(\sqrt{n}) = O(\sqrt{\lambda})$ as usual. We further choose $\ell = n\lambda = \lambda^2$. We can now choose $m = \lambda^3$, hence the condition $\ell \leq (m - \log(q) - 2\lambda)/\log(m)$ is satisfied for a sufficiently large $\lambda$ and as long as $q = \mathsf{poly}(\lambda)$. Finally, we can choose $q \geq \Omega(k^{2-\epsilon}\sigma m\ell\lambda^{1/2-c}) = \Omega(k^{2-\epsilon}\lambda^{3.5}\sigma) = O(k^{2-\epsilon}\lambda^{4})$.

The modulus-to-noise ratio for these choices will be $q/\sigma = \Omega(k^{2-\epsilon}\lambda^{3.5})$.

# 6 Vector Trapdoor Hash from DDH and LPN

In this section, we give our construction of vector trapdoor hash from DDH and the LPN assumptions.

**Building Blocks.** We need the following components for our construction:

– A PRG,

$$\mathsf{PRG} : \{0,1\}^{\lambda^{\varepsilon_{\mathsf{PRG}}}} \mapsto \mathbb{Z}_p^m$$

such that the following distributions are indistinguishable,

$$\left\{\mathsf{PRG}(\mathsf{seed}) \;\middle|\; \mathsf{seed} \leftarrow \{0,1\}^{\lambda^{\varepsilon_{\mathsf{PRG}}}}\right\}_{\lambda \in [\mathbb{N}]}$$
$$\left\{\mathsf{Ber}_{\tau(\lambda)}(\mathbb{Z}_p)^m\right\}_{\lambda \in [\mathbb{N}]}$$

Here the parameter $\tau(\lambda)$ is set such that other than with negligible probability, the Hamming weight for a sample from $\mathsf{Ber}_{\tau(\lambda)}(\mathbb{Z}_p)^m$ is $\leq m/\lambda^{1-\varepsilon_{\mathsf{wt}}}$.

The existence of such a PRG follows from the existence of any PRG such as those based on DDH. The underlying PRG output can be used as the input to the sampler for $\mathsf{Ber}_{\tau(\lambda)}(\mathbb{Z}_p)^m$.

– A balanced and blockwise resilient Boolean function (Theorem 1) $F : \{0,1\}^{\rho_{\mathsf{blk}} \cdot \rho_{\mathsf{num}}} \to \{0,1\}$.

– A randomness extractor $\mathsf{Ext} : \{0,1\}^{\lambda} \times \mathbb{G} \to \{0,1\}$. We denote $\mathsf{Ext}_S(\cdot)$ as the function that takes a group element as input and uses $S$ as the seed.

**Parameters.** We specify here the parameters, and the relationships between them necessary for our proof. The security parameter is $\lambda$. We primarily work over $\mathbb{Z}_p$, with prime order groups $\mathbb{G}$ of size $p$ with generator $g$. The reader may disregard the relationship between the parameters for now to focus on the construction's description and can return here later for the security proof.

- The group elements are represented by $\lambda_{\mathbb{G}}$ bits.
- The number of parallel repetitions be denoted by $\rho = \rho_{\mathsf{blk}} \cdot \rho_{\mathsf{num}} = \lambda^{1+\varepsilon_{\mathsf{rep}}}$. Here $\rho_{\mathsf{blk}} = \lambda^{1+\varepsilon_{\mathsf{blk}}}$ and $\rho_{\mathsf{num}} = \lambda^{\varepsilon_{\mathsf{num}}}$, thus $\varepsilon_{\mathsf{rep}} = \varepsilon_{\mathsf{blk}} + \varepsilon_{\mathsf{num}}$.
- The LPN matrices are of the form $\mathbb{Z}_p^{\lambda \times m}$, where $m = \lambda^{1+\varepsilon_{\mathsf{LPN}}}$ for $\varepsilon_{\mathsf{LPN}} \in (0, 1)$ and $\varepsilon_{\mathsf{rep}} < \varepsilon_{\mathsf{LPN}}$.
- The Bernoulli distribution for the PRG has already been set to be such that the Hamming weight of the output is $m/\lambda^{1-\varepsilon_{\mathsf{wt}}}$
- The Bernoulli distribution for the LPN error is set to be $\tau_{\mathsf{err}} = 1/\lambda^{\varepsilon_{\mathsf{err}}}$, where $1 > \varepsilon_{\mathsf{err}} > \varepsilon_{\mathsf{LPN}} + \varepsilon_{\mathsf{wt}} + \varepsilon_{\mathsf{num}}$. And $\varepsilon_{\mathsf{PRG}} < 1 + \varepsilon_{\mathsf{blk}} - (\varepsilon_{\mathsf{err}} - \varepsilon_{\mathsf{LPN}} - \varepsilon_{\mathsf{wt}})$.

**Construction.** We now present our construction of the vector trapdoor hash.

```
Setup(1^λ, 1^k)

1. $\mathbb{G}, p, g \leftarrow \mathsf{Gen}(1^\lambda)$. //group represented by $\lambda^{\varepsilon_{\mathbb{G}}}$ bits
2. Sample a seed $S \leftarrow \{0,1\}^\lambda$.
3. $\forall i \in [k]$,
    (a) Sample $\mathbf{B}_i \leftarrow \mathbb{Z}_p^{\lambda \times \lambda}$, $\mathbf{W}_i \in \mathbb{Z}_p^{\lambda \times m}$.
    (b) Set $\mathbf{G}_i := g^{\mathbf{B}_i}$.
    (c) Set $\mathbf{A}_i := \mathbf{B}_i \mathbf{W}_\mathbf{i}$. // $\in \mathbb{Z}^{\lambda \times m}$
    (d) Set $\mathbf{H}_i := g^{\mathbf{A}_i} (= \mathbf{G}_i^{\mathbf{W}_i})$.
    (e) For each repetition $\ell \in [\rho]$
        i. Sample $\mathbf{s}_i^{(\ell)} \leftarrow \mathbb{Z}_p^\lambda$.
        ii. Sample $\mathbf{e}_i^{(\ell)} \leftarrow \mathsf{Ber}_{\tau_{\mathsf{err}}}(\mathbb{Z}_p)^m$.
        iii. Set
            – $\forall j \in [k] \setminus \{i\}$, $\mathsf{ek}_{i,j}^{(\ell)} := g^{(\mathbf{s}_i^{(\ell)})^\top \mathbf{B}_j}$ // $\in \mathbb{G}^{1 \times \lambda}$
            – $\mathsf{ek}_{i,i}^{(\ell)} := g^{(\mathbf{s}_i^{(\ell)})^\top \mathbf{B}_i \mathbf{W}_\mathbf{i} + (\mathbf{e}_i^{(\ell)})^\top} (= g^{(\mathbf{s}_i^{(\ell)})^\top \mathbf{A}_i + (\mathbf{e}_i^{(\ell)})^\top})$ // $\in \mathbb{G}^{1 \times m}$
    (f) Set $\mathsf{ek}_i := (S, \{\mathsf{ek}_{i,j}^{(\ell)}\}_{j \in [k], \ell \in [\rho]})$, $\mathsf{td}_i := (S, \{\mathbf{s}_i^{(\ell)}\}_{\ell \in [\rho]})$.
4. Set $\mathsf{hk} = (g, \{\mathbf{G}_i, \mathbf{W}_i, \mathsf{ek}_{i,j}^{(\ell)}\}_{j \in [k], \ell \in [\rho]}\}_{i \in [k]})$.
5. Output $(\mathsf{hk}, \{\mathsf{ek}_i, \mathsf{td}_i\}_{i \in [k]})$.
```

```
Hash(hk, x ∈ {0,1}^{k·ε_PRG})

1. Parse $\mathsf{hk} = (g, \{\mathbf{G}_i, \mathbf{W}_i, \{\mathsf{ek}_{i,j}^{(\ell)}\}_{j \in [k], \ell \in [\rho]}\}_{i \in [k]})$.
```

```
2. Parse $\mathbf{x} = (\mathsf{seed}_1, \cdots, \mathsf{seed}_k)$.
3. Set $\mathbf{x}_i \coloneqq \mathsf{PRG}(\mathsf{seed}_i)$.
4. Set $\mathbf{h} \coloneqq \prod_{i\in[k]} \mathbf{H}_i^{\mathbf{x}_i}$ // $\in \mathbb{G}^{1\times\lambda}$ $(\mathbf{h}[\ell] = \prod_{i=1}^{k}\prod_{j=1}^{m} g^{\mathbf{A}_i[\ell,j]\mathbf{x}_i[j]})$.
5. $\forall i \in [k]$,
    (a) Set $\mathbf{y}_i \coloneqq \mathbf{W}_i\mathbf{x}_i$ //$\in \mathbb{Z}_p^{\lambda}$.
    (b) Set $\boldsymbol{\pi}_i \coloneqq (\mathbf{y}_1, \ldots, \mathbf{y}_{i-1}, \mathsf{seed}_i, \mathbf{y}_{i+1}, \ldots, \mathbf{y}_k)$ //$\in \mathbb{Z}_p^{(k-1)\lambda+m}$.
6. Output $(\mathbf{h}, \{\boldsymbol{\pi}_i\}_{i\in[k]})$.
```

$\underline{\mathsf{Encode}(\mathbf{ek}_i, \boldsymbol{\pi}_i)}$

```
1. Parse $\mathbf{ek}_i = (S, \{\mathbf{ek}_{i,j}^{(\ell)}\}_{j\in[k],\ell\in[\rho]})$.
2. Parse $\boldsymbol{\pi}_i = (\mathbf{y}_1, \ldots, \mathbf{y}_{i-1}, \mathsf{seed}_i, \mathbf{y}_{i+1}, \ldots, \mathbf{y}_k)$.
3. Set $\mathbf{x}_i \coloneqq \mathsf{PRG}(\mathsf{seed}_i)$.
4. For all $\ell \in [\rho]$, set $e_i^{(\ell)} \coloneqq \mathsf{Ext}_S\left((\mathbf{ek}_{i,i}^{(\ell)})^{\mathbf{x}_i}\prod_{j\neq i}(\mathbf{ek}_{i,j}^{(\ell)})^{\mathbf{y}_j}\right)$. //Ext is a randomness extractor.
5. Output $e_i = F(e_i^{(1)}, \ldots, e_i^{(\rho)})$.
```

$\underline{\mathsf{Decode}(\mathsf{td}_i, \mathbf{h})}$

```
1. Parse $\mathsf{td}_i = (S, \{\mathbf{s}_i^{(\ell)}\}_{\ell\in[\rho]})$.
2. For all $\ell \in [\rho]$, set $d_i^{(\ell)} \coloneqq \mathsf{Ext}_S\left(\mathbf{h}^{(\mathbf{s}_i^{(\ell)})^\top}\right)$.
3. Output $d_i = F(d_i^{(1)}, \ldots, d_i^{(\rho)})$.
```

$\underline{\mathsf{Verify}(\mathsf{hk}, \mathbf{h}, i, \boldsymbol{\pi}_i)}$

```
1. Parse $\mathsf{hk} = (g, \{\mathbf{G}_i, \mathbf{W}_i, \{\mathbf{ek}_{i,j}^{(\ell)}\}_{j\in[k],\ell\in[\rho]}\}_{i\in[k]})$.
2. Parse $\boldsymbol{\pi}_i = (\mathbf{y}_1, \ldots, \mathbf{y}_{i-1}, \mathsf{seed}_i, \mathbf{y}_{i+1}, \ldots, \mathbf{y}_k)$.
3. Set $\mathbf{H}_i \coloneqq \mathbf{G}_i^{\mathbf{W}_i}$
4. Set $\mathbf{x}_i = \mathsf{PRG}(\mathsf{seed}_i)$.
5. Output 0 if $\mathbf{h} = g^{\mathbf{0}}$.
6. Output 1 if $\mathbf{h} = \mathbf{H}_i^{\mathbf{x}_i}\prod_{j\neq i}\mathbf{G}_j^{\mathbf{y}_j}$.
```

**Lemma 11 (Completeness and Succinctness).** *The scheme presented above satisfies completeness and succinctness.*

*Proof.* Succinctness of the scheme follows directly from the choice of parameters, where the hash $\mathbf{h}$ is $\lambda$ group elements with the group elements represented by $\lambda^{\varepsilon_{\mathbb{G}}}$. Thus the total size of the hash is $\mathsf{poly}(\lambda)$.

For completeness, we simply need to argue that for an honestly generated hash $\mathbf{h}$, the probability that $\mathbf{h} = g^{\mathbf{0}}$ is negligible in $\lambda$. This is the only instance that the Verify algorithm will reject since it otherwise simply recomputes $\mathbf{h}$.

We rewrite $\mathbf{h} = \prod_{i \in [k]} \mathbf{H}_i^{\mathbf{x}_i}$ as $g^{\sum_{i \in [k]} \mathbf{A}_i \mathbf{x}_i}$. Since in the honest setting, all $\mathbf{x}_i$ are computed independently of hk and thus independently of all the $\mathbf{A}_i$, we simply need to bound the probability that for all fixed $\{\mathbf{x}_i\}_{i \in [k]}$, $\Pr\left[\sum_{i \in [k]} \mathbf{A}_i \mathbf{x}_i = \mathbf{0}\right]$ where the probability is over the choice of $\mathbf{A}_i$. We separate this into two cases: (i) $\mathbf{x_i} = \mathbf{0}$ for *every* $i$ – since $\Pr\left[\mathsf{Ber}_{\tau(\lambda)}(\mathbb{Z}_p)^m = \mathbf{0}\right] = \mathsf{negl}(\lambda)$ by our choice of parameters, by the pseudorandomness of the PRG we have that $\Pr[(\mathbf{x}_1, \ldots, \mathbf{x}_k) = \mathbf{0}] = \mathsf{negl}(\lambda)$ where each $\mathbf{x}_i$ is the output of the PRG on a random seed; and (ii) exists $i^*, j^*$ such that $\mathbf{x}_i^*[j^*] \neq 0$ – we rewrite $\Pr\left[\sum_{i \in [k]} \mathbf{A}_i \mathbf{x}_i = \mathbf{0}\right]$ as

$$\Pr\left[\forall \ell \in [\lambda] : \sum_{i \in [k]} \sum_{j \in [m]} \mathbf{A}_i[\ell, j] \mathbf{x}_i[j] = 0\right] \leq 1/p^{\ell}$$

where the bound comes from the fact that for all $\ell$ we can represent $\mathbf{A}_i^*[\ell, j^*] \in \mathbb{Z}_p$ as $1/\mathbf{x}_i^*[j^*](\sum_{i \neq i^*} \sum_{j \in [m]} \mathbf{A}_i[\ell, j]\mathbf{x}_i[j])$. □

## 6.1 Hiding

**Lemma 12.** *The scheme presented above satisfies hiding based the security of the PRG, and of the DDH and LPN assumptions.*

*Proof.* We first show that there exists an alternate algorithm $\mathsf{Setup}^*$ such that the mode indistinguishability and pseudorandomness as defined in Definition 12.

**Mode Indistinguishability.** At a high level, $\mathsf{Setup}^*$ will replace each $\mathsf{ek}_i$ by random group elements. Note that matrices $\{\mathbf{G}_i, \mathbf{W}_i\}_{i \in [k]}$ are sampled at random, and given these matrices, each $\mathsf{ek}_i$ is computed independently from other. Thus, it will suffice to present an index specific alternate setup algorithm $\mathsf{Setup}^*_{i^*}$ that only replaces $\mathsf{ek}_{i^*}$ by random. We will then show that $\mathsf{Setup}^*_{i^*}$ is indistinguishable from Setup. From the independence of generation of $\mathsf{ek}_i$s, the aforementioned indistinguishability is sufficient to establish mode indistinguishability for $\mathsf{Setup}^*$ where each $\mathsf{ek}_i$ is generated at random.

We present our proof for security via a sequence of hybrids described below. We start with the output of Setup (excluding the trapdoor), and over a sequence of hybrids replace $\mathsf{ek}_{i^*}$ by random group elements. The setup algorithm in the final hybrid will correspond to $\mathsf{Setup}^*_{i^*}$.

The output of each hybrid is $(\mathsf{hk}, \{\mathsf{ek}_i\})$.

$\mathsf{Hyb}_0$: The first hybrid generates the distribution as in the standard seup Setup. Specifically,

1. Sample $S \leftarrow \{0,1\}^{\lambda}$.
2. $\forall i \in [k]$,
   (a) Sample $\mathbf{B}_i \leftarrow \mathbb{Z}_p^{\lambda \times \lambda}$, $\mathbf{W}_i \in \mathbb{Z}_p^{\lambda \times m}$.

(b) Set $\mathbf{G}_i := g^{\mathbf{B}_i}$, $\mathbf{A}_i := \mathbf{B}_i\mathbf{W_i}$, $\mathbf{H}_i := g^{\mathbf{A}_i}$.

(c) For each repetition $\ell \in [\rho]$

i. Sample $\mathbf{s}_i^{(\ell)} \leftarrow \mathbb{Z}_p^\lambda$, $\mathbf{e}_i^{(\ell)} \leftarrow \mathsf{Ber}_{\tau_{\mathsf{err}}}(\mathbb{Z}_p)^m$.

ii. Set $\mathsf{ek}_{i,i}^{(\ell)} := g^{(\mathbf{s}_i^{(\ell)})^\top \mathbf{B}_i\mathbf{W_i}+(\mathbf{e}_i^{(\ell)})^\top}$; $\forall j \in [k] \setminus \{i\}$, $\mathsf{ek}_{i,j}^{(\ell)} := g^{(\mathbf{s}_i^{(\ell)})^\top \mathbf{B}_j}$.

(d) $\mathsf{ek}_i = (S, \{\mathsf{ek}_{i,j}^{(\ell)}\}_{j\in[k],\ell\in[\rho]})$.

3. Set $\mathsf{hk} = (g, \{\mathbf{G}_i, \mathbf{W}_i, \mathsf{ek}_{i,j}^{(\ell)}\}_{j\in[k],\ell\in[\rho]}\}_{i\in[k]})$.

$\mathsf{Hyb}_1$**:** In this hybrid, we rely on the DDH assumption to remove the use of $\mathbf{s}_{i^*}^{(1)}, \cdots, \mathbf{s}_{i^*}^{(\rho)}$ from the hybrids. We do so by changing how $\{\mathsf{ek}_{i^*,j}^{(\ell)}\}_{j\in[k]}$ is computed for each $\ell \in [\rho]$.

1. Sample $S \leftarrow \{0,1\}^\lambda$.
2. $\forall i \in [k]$,

(a) Sample $\mathbf{B}_i \leftarrow \mathbb{Z}_p^{\lambda\times\lambda}$, $\mathbf{W}_i \in \mathbb{Z}_p^{\lambda\times m}$.

(b) Set $\mathbf{G}_i := g^{\mathbf{B}_i}$, $\mathbf{A}_i := \mathbf{B}_i\mathbf{W_i}$, $\mathbf{H}_i := g^{\mathbf{A}_i}$.

3. $\forall i \neq i^*$,

(a) For each repetition $\ell \in [\rho]$

i. Sample $\mathbf{s}_i^{(\ell)} \leftarrow \mathbb{Z}_p^\lambda$, $\mathbf{e}_i^{(\ell)} \leftarrow \mathsf{Ber}_{\tau_{\mathsf{err}}}(\mathbb{Z}_p)^m$.

ii. Set $\mathsf{ek}_{i,i}^{(\ell)} := g^{(\mathbf{s}_i^{(\ell)})^\top \mathbf{B}_i\mathbf{W_i}+(\mathbf{e}_i^{(\ell)})^\top}$; $\forall j \in [k] \setminus \{i\}$, $\mathsf{ek}_{i,j}^{(\ell)} := g^{(\mathbf{s}_i^{(\ell)})^\top \mathbf{B}_j}$.

4. For each repetition $\ell \in [\rho]$

(a) $\underline{\text{Sample } \mathbf{e}_{i^*}^{(\ell)} \leftarrow \mathsf{Ber}_{\tau_{\mathsf{err}}}(\mathbb{Z}_p)^m, \text{ and } \forall j \in [k]\ \mathbf{t}_{i^*,j}^{(\ell)} \leftarrow \mathbb{Z}_p^\lambda.}$

(b) $\underline{\text{Set } \mathsf{ek}_{i^*,i^*}^{(\ell)} := g^{(\mathbf{t}_{i^*,i^*}^{(\ell)})^\top \mathbf{W_{i^*}}+(\mathbf{e}_{i^*}^{(\ell)})^\top}; \forall j \in [k] \setminus \{i^*\}, \mathsf{ek}_{i^*,j}^{(\ell)} := g^{(\mathbf{t}_{i^*,j}^{(\ell)})^\top}.}$

5. $\forall i \in [k]$, $\mathsf{ek}_i = (S, \{\mathsf{ek}_{i,j}^{(\ell)}\}_{j\in[k],\ell\in[\rho]})$.
6. Set $\mathsf{hk} = (g, \{\mathbf{G}_i, \mathbf{W}_i, \mathsf{ek}_{i,j}^{(\ell)}\}_{j\in[k],\ell\in[\rho]}\}_{i\in[k]})$.

$\underline{\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_0}$**.** We start by defining a sequence $\rho$ sub-hybrids, $\mathsf{Hyb}_{1,0}, \ldots, \mathsf{Hyb}_{1,\rho}$. In $\mathsf{Hyb}_{1,\ell}$ we change only the computation of $\{\mathsf{ek}_{i^*,j}^{(\ell)}\}_{j\in[k]}$ from

$$\mathsf{ek}_{i^*,i^*}^{(\ell)} := g^{(\mathbf{s}_{i^*}^{(\ell)})^\top \mathbf{B}_{i^*}\mathbf{W_{i^*}}+(\mathbf{e}_{i^*}^{(\ell)})^\top}; \forall j \in [k] \setminus \{i^*\},\ \mathsf{ek}_{i^*,j}^{(\ell)} := g^{(\mathbf{s}_{i^*}^{(\ell)})^\top \mathbf{B}_j}$$

to

$$\mathsf{ek}_{i^*,i^*}^{(\ell)} := g^{(\mathbf{t}_{i^*,i^*}^{(\ell)})^\top \mathbf{W_i}+(\mathbf{e}_{i^*}^{(\ell)})^\top}; \forall j \in [k] \setminus \{i^*\}, \mathsf{ek}_{i^*,j}^{(\ell)} := g^{(\mathbf{t}_{i^*,j}^{(\ell)})^\top}.$$

Thus, $\mathsf{Hyb}_{1,0} \equiv \mathsf{Hyb}_0$ and $\mathsf{Hyb}_{1,\rho} \equiv \mathsf{Hyb}_1$.

For every $\ell$, we show that $\mathsf{Hyb}_{1,\ell}$ and $\mathsf{Hyb}_{1,\ell+1}$ are indistinguishable based on the DDH assumption. In particular, for every $\ell \in [\rho]$ we rely on the matrix DDH assumption which states that the following distributions are computationally indistinguishable:

$$g^{\mathbf{B}_1}, \cdots, g^{\mathbf{B}_k}, g^{(\mathbf{s}_{i^*}^{(\ell)})^\top \mathbf{B}_1} \cdots g^{(\mathbf{s}_{i^*}^{(\ell)})^\top \mathbf{B}_k} \text{ and } g^{\mathbf{B}_1}, \cdots, g^{\mathbf{B}_k}, g^{(\mathbf{t}_{i^*,1}^{(\ell)})^\top} \cdots g^{(\mathbf{t}_{i^*,k}^{(\ell)})^\top},$$

where $\mathbf{s}_{i^*}^{(\ell)}, \{\mathbf{B}_j, \mathbf{t}_{i^*,j}^{(\ell)}\}_{j\in[k]}$ are drawn at random (with appropriate dimensions) from $\mathbb{Z}_p$.

In the reduction to the above assumption, note that $\mathsf{ek}_{i^*,i^*}^{(\ell)}$ can be computed given either $g^{(\mathbf{t}_{i^*,i^*}^{(\ell)})^\top}$ or $g^{(\mathbf{s}_{i^*}^{(\ell)})^\top \mathbf{B}_{i^*}}$ since $\mathbf{W}_{i^*}$ and $\mathbf{e}_{i^*}^{(\ell)}$ are known to the reduction in the clear. Further, the computation of $\{\mathsf{ek}_{i^*,j}^{(\ell')}\}_{j\in[k]}$ for $\ell' \neq \ell$ can be computed directly using $g^{\mathbf{B}_1}, \cdots, g^{\mathbf{B}_k}$ and without knowledge of $\mathbf{s}_{i^*}^{(\ell)}$.

$\mathsf{Hyb}_2$: In this hybrid, we rely on LPN to replace the noisy terms in $\{\mathsf{ek}_{i^*,j}^{(\ell)}\}_{j\in[k]}$ for each $\ell \in [\rho]$ with a random group element.

```
1. Sample $S \leftarrow \{0,1\}^\lambda$.
2. $\forall i \in [k]$,
   (a) Sample $\mathbf{B}_i \leftarrow \mathbb{Z}_p^{\lambda\times\lambda}$, $\mathbf{W}_i \in \mathbb{Z}_p^{\lambda\times m}$.
   (b) Set $\mathbf{G}_i := g^{\mathbf{B}_i}$, $\mathbf{A}_i := \mathbf{B}_i\mathbf{W_i}$, $\mathbf{H}_i := g^{\mathbf{A}_i}$.
   (c) Set $\mathsf{ek}_i := (S, \{\mathsf{ek}_{i,j}^{(\ell)}\}_{j\in[k],\ell\in[\rho]})$.
3. $\forall i \neq i^*$,
   (a) For each repetition $\ell \in [\rho]$
      i. Sample $\mathbf{s}_i^{(\ell)} \leftarrow \mathbb{Z}_p^\lambda$, $\mathbf{e}_i^{(\ell)} \leftarrow \mathsf{Ber}_{\tau_{\mathsf{err}}}(\mathbb{Z}_p)^m$.
      ii. Set $\mathsf{ek}_{i,i}^{(\ell)} := g^{(\mathbf{s}_i^{(\ell)})^\top \mathbf{B}_i \mathbf{W_i} + (\mathbf{e}_i^{(\ell)})^\top}$; $\forall j \in [k]\setminus\{i\}$, $\mathsf{ek}_{i,j}^{(\ell)} := g^{(\mathbf{s}_i^{(\ell)})^\top \mathbf{B}_j}$.
4. For each repetition $\ell \in [\rho]$
   (a) Sample $\mathbf{v}_{i^*}^{(\ell)} \leftarrow \mathbb{Z}_p^m$, $\forall j \neq i^*$ $\mathbf{t}_{i^*,j}^{(\ell)} \leftarrow \mathbb{Z}_p^\lambda$.
   (b) Set $\mathsf{ek}_{i^*,i^*}^{(\ell)} := g^{(\mathbf{v}_{i^*}^{(\ell)})^\top}$; $\forall j \in [k]\setminus\{i^*\}$, $\mathsf{ek}_{i^*,j}^{(\ell)} := g^{(\mathbf{t}_{i^*,j}^{(\ell)})^\top}$.
5. $\forall i \in [k]$, $\mathsf{ek}_i = (S, \{\mathsf{ek}_{i,j}^{(\ell)}\}_{j\in[k],\ell\in[\rho]})$.
6. Set $\mathsf{hk} = (g, \{\mathbf{G}_i, \mathbf{W}_i, \mathsf{ek}_{i,j}^{(\ell)}\}_{j\in[k],\ell\in[\rho]}\}_{i\in[k]})$.
```

$\underline{\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_1}$. We rely on the LPN to argue indistinguishability of hybrids. As in the previous hybrid, we define a sequence of $\rho$ sub-hybrids, $\mathsf{Hyb}_{2,0}, \ldots, \mathsf{Hyb}_{2,\rho}$. In $\mathsf{Hyb}_{2,\ell}$ we change only the computation of $\mathsf{ek}_{i^*,i^*}^{(\ell)}$ from $g^{(\mathbf{t}_{i^*,i^*}^{(\ell)})^\top \mathbf{W_{i^*}} + (\mathbf{e}_{i^*}^{(\ell)})^\top}$ to $g^{(\mathbf{v}_{i^*}^{(\ell)})^\top}$. Thus, $\mathsf{Hyb}_{2,0} \equiv \mathsf{Hyb}_1$ and $\mathsf{Hyb}_{2,\rho} \equiv \mathsf{Hyb}_2$.

To argue indistinguishability of $\mathsf{Hyb}_{2,\ell}$ and $\mathsf{Hyb}_{2,\ell}$, we rely on the $\mathsf{LPN}_{\lambda,m,\tau}$ assumption that shows that the following two distributions are indistinguishable,

$$\mathbf{W}_{i^*}, (\mathbf{t}_{i^*,i^*}^{(\ell)})^\top \mathbf{W_{i^*}} + (\mathbf{e}_{i^*}^{(\ell)})^\top \text{ and } \mathbf{W}_{i^*}, (\mathbf{v}_{i^*}^{(\ell)})^\top$$

where $\mathbf{W}_{i^*}, \mathbf{t}_{i^*,i^*}^{(\ell)}$ and $\mathbf{v}_{i^*}^{(\ell)}$ are sampled from $\mathbb{Z}_p$ with the appropriate dimension, and $\mathbf{e}_{i^*}^{(\ell)}$ is sampled from $\mathsf{Ber}_\tau(\mathbb{Z}_p)^m$. Note that by our choice of parameters, we have that $m = \lambda^{1+\varepsilon_{\mathsf{LPN}}} > \lambda$ as required since $\mathbf{t}_{i^*,i^*}^{(\ell)} \in \mathbb{Z}_p^\lambda$.

For the reduction, given either $(\mathbf{t}_{i^*,i^*}^{(\ell)})^\top \mathbf{W_{i^*}} + (\mathbf{e}_{i^*}^{(\ell)})^\top$ or $(\mathbf{v}_{i^*}^{(\ell)})^\top$ from the LPN challenger, $\mathsf{ek}_{i^*,i^*}^{(\ell)}$ can be computed by simply raising $g$ to the challenge vector.

The output of $\mathsf{Hyb}_2$ is exactly the setup algorithm $\mathsf{Setup}_{i^*}^*$.

**Pseudorandomness.** We now show that the scheme satisfies pseudorandomness. Let us fix the challenge index to be $i^*$. In each hybrid, we shall argue that the distribution over the adversary $\mathcal{A}$'s input $(\mathsf{hk}, h, \{\mathsf{ek}_i, \pi_i\}_{i \neq i^*}, (\mathsf{ek}_{i^*}, r_{i^*}))$ remains indistinguishable, thus establishing that the scheme is secure when the starting and ending hybrids correspond to the distributions in the 'real' and 'ideal' games in the pseudorandomness security definition.

The output of each hybrid is $(\mathsf{hk}, h, \{\mathsf{ek}_i, \pi_i\}_{i \neq i^*}, (\mathsf{ek}_{i^*}, e_{i^*}))$.

$\mathsf{Hyb}_0$**:** The first hybrid generates the distribution as in the 'real' game. Here the setup parameters are sampled according to $\mathsf{Setup}^*$, where all the $\mathbf{ek}_i$ are random elements. Specifically,

1. Sample $S \leftarrow \{0,1\}^\lambda$.
2. $\forall i \in [k]$,
   (a) Sample $\mathbf{B}_i \leftarrow \mathbb{Z}_p^{\lambda \times \lambda}$, $\mathbf{W}_i \in \mathbb{Z}_p^{\lambda \times m}$.
   (b) Set $\mathbf{G}_i := g^{\mathbf{B}_i}$, $\mathbf{A}_i := \mathbf{B}_i \mathbf{W_i}$, $\mathbf{H}_i := g^{\mathbf{A}_i}$.
   (c) For each repetition $\ell \in [\rho]$
      i. Sample $\mathbf{v}_i^{(\ell)} \leftarrow \mathbb{Z}_p^m$, $\forall j \neq i$, $\mathbf{t}_{i,j}^{(\ell)} \leftarrow \mathbb{Z}_p^\lambda$.
      ii. Set $\mathbf{ek}_{i,i}^{(\ell)} := g^{(\mathbf{v}_i^{(\ell)})^\top}$; $\forall j \in [k] \setminus \{i\}$, $\mathbf{ek}_{i,j}^{(\ell)} := g^{(\mathbf{t}_{i,j}^{(\ell)})^\top}$.
   (d) Set $\mathbf{ek}_i = (S, \{\mathbf{ek}_{i,j}^{(\ell)}\}_{j \in [k], \ell \in [\rho]})$.
3. $\forall i \in [k]$, set $\mathsf{seed}_i \leftarrow \{0,1\}^{\varepsilon_{\mathsf{PRG}}}$, $\mathbf{x}_i := \mathsf{PRG}(\mathsf{seed}_i)$, $\mathbf{y}_i := \mathbf{W}_i \mathbf{x}_i$.
4. $\forall i \neq i^*$, $\boldsymbol{\pi}_i := (\mathbf{y}_1, \ldots, \mathbf{y}_{i-1}, \mathsf{seed}_i, \mathbf{y}_{i+1}, \ldots, \mathbf{y}_k)$.
5. For all $\ell \in [\rho]$, set $e_{i^*}^{(\ell)} := \mathsf{Ext}_S(g^{(\mathbf{v}_{i^*}^{(\ell)})^\top \mathbf{x}_{i^*}} \prod_{j \neq i^*} (\mathbf{ek}_{i^*,j}^{(\ell)})^{\mathbf{y}_j})$.
6. Set $e_{i^*} = F(e_{i^*}^{(1)}, \ldots, e_{i^*}^{(\rho)})$.
7. Set $\mathsf{hk} = (g, \{\mathbf{G}_i, \mathbf{W}_i, \mathbf{ek}_{i,j}^{(\ell)}\}_{j \in [k], \ell \in [\rho]}\}_{i \in [k]})$.

Note that in the computation of $e_{i^*}^{(\ell)}$, we make a syntactic change and replace $e_{i^*}^{(\ell)} := \mathsf{Ext}_S((\mathbf{ek}_{i^*,i^*}^{(\ell)})^{\mathbf{x}_{i^*}} \prod_{j \neq i^*} (\mathbf{ek}_{i^*,j}^{(\ell)})^{\mathbf{y}_j})$ with $e_{i^*}^{(\ell)} := \mathsf{Ext}_S(g^{(\mathbf{v}_{i^*}^{(\ell)})^\top \mathbf{x}_{i^*}} \prod_{j \neq i^*} (\mathbf{ek}_{i^*,j}^{(\ell)})^{\mathbf{y}_j})$. This change will be useful for arguing security in subsequent hybrids.

$\mathsf{Hyb}_1$**:** In this hybrid, we sample $\mathbf{x}_{i^*}$ randomly instead of using the output of PRG. Specifically,

1. Sample $S \leftarrow \{0,1\}^\lambda$.
2. $\forall i \in [k]$,
   (a) Sample $\mathbf{B}_i \leftarrow \mathbb{Z}_p^{\lambda \times \lambda}$, $\mathbf{W}_i \in \mathbb{Z}_p^{\lambda \times m}$.
   (b) Set $\mathbf{G}_i := g^{\mathbf{B}_i}$, $\mathbf{A}_i := \mathbf{B}_i \mathbf{W_i}$, $\mathbf{H}_i := g^{\mathbf{A}_i}$.
   (c) For each repetition $\ell \in [\rho]$
      i. Sample $\mathbf{v}_i^{(\ell)} \leftarrow \mathbb{Z}_p^m$, $\forall j \neq i$, $\mathbf{t}_{i,j}^{(\ell)} \leftarrow \mathbb{Z}_p^\lambda$.
      ii. Set $\mathbf{ek}_{i,i}^{(\ell)} := g^{(\mathbf{v}_i^{(\ell)})^\top}$; $\forall j \in [k] \setminus \{i\}$, $\mathbf{ek}_{i,j}^{(\ell)} := g^{(\mathbf{t}_{i,j}^{(\ell)})^\top}$.
   (d) Set $\mathbf{ek}_i = (S, \{\mathbf{ek}_{i,j}^{(\ell)}\}_{j \in [k], \ell \in [\rho]})$.
3. $\forall i \neq i^*$, set $\mathsf{seed}_i \leftarrow \{0,1\}^{\varepsilon_{\mathsf{PRG}}}$, $\mathbf{x}_i := \mathsf{PRG}(\mathsf{seed}_i)$, $\mathbf{y}_i := \mathbf{W}_i \mathbf{x}_i$.
4. $\underline{\mathbf{x}_{i^*} \leftarrow \mathsf{Ber}_{\tau(\lambda)}(\mathbb{Z}_p)^m,\ \mathbf{y}_{i^*} := \mathbf{W}_{i^*} \mathbf{x}_{i^*}.}$
5. $\forall i \neq i^*$, $\boldsymbol{\pi}_i := (\mathbf{y}_1, \ldots, \mathbf{y}_{i-1}, \mathsf{seed}_i, \mathbf{y}_{i+1}, \ldots, \mathbf{y}_k)$.
6. For all $\ell \in [\rho]$, set $e_{i^*}^{(\ell)} := \mathsf{Ext}_S(g^{(\mathbf{v}_{i^*}^{(\ell)})^\top \mathbf{x}_{i^*}} \prod_{j \neq i^*} (\mathbf{ek}_{i^*,j}^{(\ell)})^{\mathbf{y}_j})$.

7. Set $e_{i^*} = F(e_{i^*}^{(1)}, \ldots, e_{i^*}^{(\rho)})$.
8. Set $\mathsf{hk} = (g, \{\mathbf{G}_i, \mathbf{W}_i, \mathsf{ek}_{i,j}^{(\ell)}\}_{j\in[k],\ell\in[\rho]}\}_{i\in[k]})$.

$\underline{\mathsf{Hyb}_1 \approx_c \mathsf{Hyb}_0}$. . This follows from the indistinguishability of the pseudorandom generator PRG. We rely here on the fact that $\mathsf{seed}_{i^*}$ is not used anywhere else in the two hybrids outside of the computation of $\mathbf{x}_{i^*}$. In particular, $\boldsymbol{\pi}_{i^*}$, which requires $\mathsf{seed}_{i^*}$ is never computed or given out to the adversary.

$\mathsf{Hyb}_2$: In this hybrid, we rely on the dual LPN to remove $\mathbf{x}_{i^*}$ from the experiment.

```
1. Sample $S \leftarrow \{0,1\}^\lambda$.
2. $\forall i \in [k]$,
   (a) Sample $\mathbf{B}_i \leftarrow \mathbb{Z}_p^{\lambda\times\lambda}$, $\mathbf{W}_i \in \mathbb{Z}_p^{\lambda\times m}$.
   (b) Set $\mathbf{G}_i := g^{\mathbf{B}_i}$, $\mathbf{A}_i := \mathbf{B}_i\mathbf{W}_\mathbf{i}$, $\mathbf{H}_i := g^{\mathbf{A}_i}$.
   (c) For each repetition $\ell \in [\rho]$
       i. Sample $\mathbf{v}_i^{(\ell)} \leftarrow \mathbb{Z}_p^m$, $\forall j \neq i$, $\mathbf{t}_{i,j}^{(\ell)} \leftarrow \mathbb{Z}_p^\lambda$.
       ii. Set $\mathsf{ek}_{i,i}^{(\ell)} := g^{(\mathbf{v}_i^{(\ell)})^\top}$; $\forall j \in [k]\setminus\{i\}$, $\mathsf{ek}_{i,j}^{(\ell)} := g^{(\mathbf{t}_{i,j}^{(\ell)})^\top}$.
   (d) Set $\mathsf{ek}_i = (S, \{\mathsf{ek}_{i,j}^{(\ell)}\}_{j\in[k],\ell\in[\rho]})$.
3. $\forall i \neq i^*$, set $\mathsf{seed}_i \leftarrow \{0,1\}^{\varepsilon_{\mathsf{PRG}}}$, $\mathbf{x}_i := \mathsf{PRG}(\mathsf{seed}_i)$, $\mathbf{y}_i := \mathbf{W}_i\mathbf{x}_i$.
4. $\underline{\mathbf{y}_{i^*} \leftarrow \mathbb{Z}_p^\lambda}$.
5. $\forall i \neq i^*$, $\boldsymbol{\pi}_i := (\mathbf{y}_1, \ldots, \mathbf{y}_{i-1}, \mathsf{seed}_i, \mathbf{y}_{i+1}, \ldots, \mathbf{y}_k)$.
6. $\underline{\text{For all } \ell \in [\rho] \text{, sample } r_{i^*}^{(\ell)} \leftarrow \mathbb{Z}_p \text{ and set } e_{i^*}^{(\ell)} := \mathsf{Ext}_S(g^{r_{i^*}^{(\ell)}} \prod_{j\neq i^*} (\mathsf{ek}_{i^*,j}^{(\ell)})^{\mathbf{y}_j})}$.
7. Set $e_{i^*} = F(e_{i^*}^{(1)}, \ldots, e_{i^*}^{(\rho)})$.
8. Set $\mathsf{hk} = (g, \{\mathbf{G}_i, \mathbf{W}_i, \mathsf{ek}_{i,j}^{(\ell)}\}_{j\in[k],\ell\in[\rho]}\}_{i\in[k]})$.
```

$\underline{\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_1}$. We rely on the dual LPN to show the indistinguishability of the two hybrids. Unlike in the previous hybrids, we will *not* define sub-hybrids and will have to argue security 'in one shot'. Specifically, the dual LPN assumption lets us argue the following distributions are indistinguishable

$$\mathbf{W}_{i^*}, (\mathbf{v}_{i^*}^{(1)})^\top, \ldots, (\mathbf{v}_{i^*}^{(\rho)})^\top, \mathbf{W}_{i^*}\mathbf{x}_{i^*}, (\mathbf{v}_{i^*}^{(1)})^\top\mathbf{x}_{i^*} \ldots, (\mathbf{v}_{i^*}^{(\rho)})^\top\mathbf{x}_{i^*}$$
$$\text{and } \mathbf{W}_{i^*}, (\mathbf{v}_{i^*}^{(1)})^\top, \ldots, (\mathbf{v}_{i^*}^{(\rho)})^\top, \mathbf{y}_{i^*}, r_{i^*}^{(1)} \ldots, r_{i^*}^{(\rho)}$$

where $\mathbf{W}_{i^*} \in \mathbb{Z}_p^{\lambda\times m}, \mathbf{v}_{i^*}^{(1)}, \ldots, \mathbf{v}_{i^*}^{(\rho)} \in \mathbb{Z}_p^m, \mathbf{y}_{i^*} \in \mathbb{Z}_p^\lambda, r_{i^*}^{(1)} \ldots, r_{i^*}^{(\rho)} \in \mathbb{Z}_p$ and $\mathbf{x}_{i^*} \leftarrow \mathsf{Ber}_{\tau(\lambda)}(\mathbb{Z}_p)^m$. We view the dual LPN matrix to be $(\mathbf{W}_{i^*}, (\mathbf{v}_{i^*}^{(1)})^\top, \ldots, (\mathbf{v}_{i^*}^{(\rho)})^\top)^\top \in \mathbb{Z}_p^{m\times(\lambda+\rho)}$. Thus to rely on the dual LPN, we need $\lambda + \rho << m$. By our choice of parameters, we have $\rho = \lambda^{1+\varepsilon_{\mathsf{rep}}}$ and $m = \lambda^{1+\varepsilon_{\mathsf{LPN}}}$ where $\varepsilon_{\mathsf{rep}} < \varepsilon_{\mathsf{LPN}}$ which enables to use the dual LPN assumption. The indistinguishability directly follows by simply raising the appropriate values to $g$,

$\mathsf{Hyb}_3$: In this hybrid, we change $e_i$ to be a randomly sampled bit.

```
1. Sample $S \leftarrow \{0,1\}^\lambda$.
```

2. $\forall i \in [k]$,
   (a) Sample $\mathbf{B}_i \leftarrow \mathbb{Z}_p^{\lambda\times\lambda}$, $\mathbf{W}_i \in \mathbb{Z}_p^{\lambda\times m}$.
   (b) Set $\mathbf{G}_i := g^{\mathbf{B}_i}$, $\mathbf{A}_i := \mathbf{B}_i\mathbf{W}_\mathbf{i}$, $\mathbf{H}_i := g^{\mathbf{A}_i}$.
   (c) For each repetition $\ell \in [\rho]$
      i. Sample $\mathbf{v}_i^{(\ell)} \leftarrow \mathbb{Z}_p^m$, $\forall j \neq i$, $\mathbf{t}_{i,j}^{(\ell)} \leftarrow \mathbb{Z}_p^\lambda$.
      ii. Set $\mathsf{ek}_{i,i}^{(\ell)} := g^{(\mathbf{v}_i^{(\ell)})^\top}$; $\forall j \in [k] \setminus \{i\}$, $\mathsf{ek}_{i,j}^{(\ell)} := g^{(\mathbf{t}_{i,j}^{(\ell)})^\top}$.
   (d) Set $\mathsf{ek}_i = (S, \{\mathsf{ek}_{i,j}^{(\ell)}\}_{j\in[k],\ell\in[\rho]})$.
3. $\forall i \neq i^*$, set $\mathsf{seed}_i \leftarrow \{0,1\}^{\varepsilon_{\mathsf{PRG}}}$, $\mathbf{x}_i := \mathsf{PRG}(\mathsf{seed}_i)$, $\mathbf{y}_i := \mathbf{W}_i\mathbf{x}_i$.
4. $\mathbf{y}_{i^*} \leftarrow \mathbb{Z}_p^\lambda$.
5. $\forall i \neq i^*$, $\boldsymbol{\pi}_i := (\mathbf{y}_1, \dots, \mathbf{y}_{i-1}, \mathsf{seed}_i, \mathbf{y}_{i+1}, \dots, \mathbf{y}_k)$.
6. For all $\ell \in [\rho]$, sample $r_{i^*}^{(\ell)} \leftarrow \mathbb{Z}_p$ and set $e_{i^*}^{(\ell)} := \mathsf{Ext}_S(g^{r_{i^*}^{(\ell)}} \prod_{j\neq i^*}(\mathsf{ek}_{i^*,j}^{(\ell)})^{\mathbf{y}_j})$.
7. <u>Set $e_{i^*} \leftarrow \{0,1\}$.</u>
8. Set $\mathsf{hk} = (g, \{\mathbf{G}_i, \mathbf{W}_i, \mathsf{ek}_{i,j}^{(\ell)}\}_{j\in[k],\ell\in[\rho]}\}_{i\in[k]})$.

<u>$\mathsf{Hyb}_3 \approx_s \mathsf{Hyb}_2$.</u> This follows from the fact that: (i) each $e_{i^*}^{(\ell)}$ is statistically close to a random bit as the output of $\mathsf{Ext}_S$ is statistically close to random bit when given a random group element as input, which is case as $g^{r_{i^*}^{(\ell)}}$ is a random group element; and (ii) $F$ is a balanced function.

Since $\mathsf{Hyb}_3$ is the challenge experiment in the 'ideal world', this concludes our proof of psuedorandomness.

□

We note that although the $\mathsf{Setup}^*$ algorithm produces keys that are indistinguishable from random, the pseudorandomness only holds against computationally bounded adversaries. This results in our scheme not satisfying the dual-mode property.

## 6.2 Statistical Binding

We will prove the statistical binding property via a sequence of Lemmas.

Fix any hash $\mathbf{h} \neq 0$. For $i \in [k]$, let $\mathsf{BAD}_i$ be the event that there exists a proof $\boldsymbol{\pi}_i$ such that (i) $\boldsymbol{\pi}_i$ verifies at index $i$ for the hash $\mathbf{h}$; and (ii) $e_i$ computed using $\boldsymbol{\pi}_i$ via the encode function is different than $d_i$ computed using the trapdoor. The probability of $\mathsf{BAD}_i$ is defined over the random choices of $\mathbf{s}_i^{(\ell)}$ and $\mathbf{e}_i^{(\ell)}$ for all $\ell \in [\rho]$.

The following Lemma establishes that for any fixed index $i \in [k]$, the event $\mathsf{BAD}_i$ happens only with a inverse poly small probability.

**Lemma 13.** *There exists a constant $\delta > 0$ such that for any fixed $i \in [k]$*

$$\Pr[\mathsf{BAD}_i] \leq 1/\lambda^\delta,$$

*where the probability is defined over the random choices of $\mathbf{s}_i^{(\ell)}$ and $\mathbf{e}_i^{(\ell)}$ for all $\ell \in [\rho]$.*

Before we prove Lemma 13, we show how to establish the statistical binding property via Lemma 13.

**Theorem 6.** *The vector trapdoor-hash* $(\mathsf{Setup}, \mathsf{Hash}, \mathsf{Encode}, \mathsf{Decode}, \mathsf{Verify})$ *is statistically binding.*

*Proof.* First note that by our description, for every $i$, $\mathsf{BAD}_i$ is determined by independently chosen random coins, namely the $\mathbf{s}_i^{(\ell)}$ and $\mathbf{e}_i^{(\ell)}$. Hence the events $\mathsf{BAD}_1, \ldots, \mathsf{BAD}_k$ are independent. In the following, let $k = \lambda^\gamma$ for some constant $\gamma$. In expectation, there are $k/\lambda^\delta = \lambda^{\gamma-\delta}$ bad indices. We can now bound the number of bad indices using the Chernoff bound as the $\mathsf{BAD}_i$ are independent events. That is, it holds that

$$\Pr\left[|\{i \mid \mathsf{BAD}_i \text{ happens }\}| \geq 2\lambda^{\gamma-\delta}\right] < e^{-\lambda^{\gamma-\delta}/3}.$$

Note that while the events $\mathsf{BAD}_i$ are defined for a fixed $\mathbf{h}$, for statistical binding the above should hold for all $\mathbf{h}$. We achieve this by applying a union bound over all possible $\mathbf{h}$. Noting that there are at most $|\mathbb{G}|^\lambda = 2^{\lambda^{1+\epsilon_\mathbb{G}}}$ possible choice for $\mathbf{h}$, we get that the probability that there exists an $\mathbf{h}$ such that there are more than $2\lambda^{\gamma-\delta}$ bad indices is bounded by $e^{-\lambda^{\gamma-\delta}/3} \cdot 2^{\lambda^{1+\epsilon_\mathbb{G}}} = 2^{-O(\lambda^{\gamma-\delta})+\lambda^{1+\epsilon_\mathbb{G}}}$. This expression is negligible whenever $\gamma > \delta + 1 + \epsilon_\mathbb{G}$. Noting that we can choose $k = \lambda^\gamma$ to be an arbitrarily large polynomial, i.e. make $\gamma$ an arbitrarily large constant, we have established that our scheme is statistically binding. □ □

We will now prove Lemma 13.

*Proof of Lemma 13.* We start by re-writing the computation of $e_i^{(\ell)}$ in $\mathsf{Encode}(\mathbf{ek}_i, \boldsymbol{\pi}_i)$ where $\boldsymbol{\pi}_i = (\mathbf{y}_1, \ldots, \mathbf{y}_{i-1}, \mathsf{seed}_i, \mathbf{y}_{i+1}, \ldots, \mathbf{y}_k)$ to be

$$\begin{aligned}
e_i^{(\ell)} &= \mathsf{Ext}_S\left(\left(g^{(\mathbf{s}_i^{(\ell)})^\top \mathbf{B}_i \mathbf{W}_\mathbf{i} + (\mathbf{e}_i^{(\ell)})^\top}\right)^{\mathbf{x}_i} \prod_{j \neq i} \left(g^{(\mathbf{s}_i^{(\ell)})^\top \mathbf{B}_j}\right)^{\mathbf{y}_i}\right) \\
&= \mathsf{Ext}_S\left(g^{(\mathbf{e}_i^{(\ell)})^\top \mathbf{x}_i} \left(g^{(\mathbf{s}_i^{(\ell)})^\top \mathbf{B}_i \mathbf{W}_\mathbf{i}}\right)^{\mathbf{x}_i} \prod_{j \neq i} \left(g^{(\mathbf{s}_i^{(\ell)})^\top \mathbf{B}_j}\right)^{\mathbf{y}_i}\right) \\
&= \mathsf{Ext}_S\left(g^{(\mathbf{e}_i^{(\ell)})^\top \mathbf{x}_i} \left(g^{\mathbf{B}_i \mathbf{W}_\mathbf{i} \mathbf{x}_i} \prod_{j \neq i} \left(g^{\mathbf{B}_j}\right)^{\mathbf{y}_i}\right)^{(\mathbf{s}_i^{(\ell)})^\top}\right) \\
&= \mathsf{Ext}_S\left(g^{(\mathbf{e}_i^{(\ell)})^\top \mathbf{x}_i} \left(g^{\mathbf{A}_i \mathbf{x}_i} \prod_{j \neq i} g^{\mathbf{B}_j \mathbf{y}_i}\right)^{(\mathbf{s}_i^{(\ell)})^\top}\right) \\
&= \mathsf{Ext}_S\left(g^{(\mathbf{e}_i^{(\ell)})^\top \mathbf{x}_i} \left(\mathbf{H}_i^{\mathbf{x}_i} \prod_{j \neq i} \mathbf{G}_j^{\mathbf{y}_i}\right)^{(\mathbf{s}_i^{(\ell)})^\top}\right)
\end{aligned}$$

If $\mathsf{Verify}(\mathsf{hk}, \mathbf{h}, i, \boldsymbol{\pi}_i) = 1$, by the description of Verify we have that $\mathbf{h} = \mathbf{H}_i^{\mathbf{x}_i} \prod_{j \neq i} \mathbf{G}_j^{\mathbf{y}_i}$. Thus, if the output of the Verify algorithm is 1, we have

$$e_i^{(\ell)} = \mathsf{Ext}_S \left( g^{(\mathbf{e}_i^{(\ell)})^\top \mathbf{x}_i} \mathbf{h}^{\mathbf{s}_i^{(\ell)}} \right).$$

Since $d_i^{(\ell)} := \mathsf{Ext}_S(\mathbf{h}^{(\mathbf{s}_i^{(\ell)})^\top})$ in the Decode algorithm, it follows that $e_i^{(\ell)} \neq d_i^{(\ell)}$ if and only if $(\mathbf{e}_i^{(\ell)})^\top \mathbf{x}_i \neq 0 \mod p$.

We now argue that the probability that there exists an $\mathbf{x}_i$ such that, within any block of size $\rho_{\mathsf{blk}}$, the number of indices where $e_i^{(\ell)} \neq d_i^{(\ell)}$ exceeds a certain threshold is small.

We first bound the probability that $(\mathbf{e}_i^{(\ell)})^\top \mathbf{x}_i \neq 0 \mod p$ for any *fixed* $\mathbf{x}_i$ and $\ell$. This is done via the following claim.

**Claim 1.** *There exists a constant $\delta' > 0$ such that for any fixed $\mathbf{x}_i$ of Hamming weight $m/\lambda^{1-\varepsilon_{\mathsf{wt}}}$ and any $\ell \in [\rho]$,*

$$\Pr_{\mathbf{e}_i^{(\ell)} \leftarrow \mathsf{Ber}_\tau(\mathbb{Z}_p)^m}\left[(\mathbf{e}_i^{(\ell)})^\top \mathbf{x}_i = 0 \mod p\right] \geq 1 - \frac{1}{\lambda^{\delta'}}.$$

To prove the claim, we substitute parameters $m = \lambda^{1+\varepsilon_{\mathsf{LPN}}}$, $\tau = 1/\lambda^{\varepsilon_{\mathsf{err}}}$ and get $m/\lambda^{1-\varepsilon_{\mathsf{wt}}} = \lambda^{\varepsilon_{\mathsf{LPN}}+\varepsilon_{\mathsf{wt}}}$. By independence of the $\mathbf{e}_i^{(\ell)}$ we get that

$$\begin{aligned}\Pr_{\mathbf{e}_i^{(\ell)} \leftarrow \mathsf{Ber}_\tau(\mathbb{Z}_p)^m}\left[(\mathbf{e}_i^{(\ell)})^\top \mathbf{x}_i = 0 \mod p\right] &\geq \left(1 - \frac{1}{\lambda^{\varepsilon_{\mathsf{err}}}}\right)^{\lambda^{\varepsilon_{\mathsf{LPN}}+\varepsilon_{\mathsf{wt}}}} \\ &\geq 1 - \frac{1}{\lambda^{\varepsilon_{\mathsf{err}}-\varepsilon_{\mathsf{LPN}}-\varepsilon_{\mathsf{wt}}}}.\end{aligned}$$

Since $\varepsilon_{\mathsf{err}} > \varepsilon_{\mathsf{LPN}} + \varepsilon_{\mathsf{wt}}$, we complete the proof of the claim by setting $\delta' = \varepsilon_{\mathsf{err}} - \varepsilon_{\mathsf{LPN}} - \varepsilon_{\mathsf{wt}}$.

From this claim, we get that for any block of size $\rho_{\mathsf{blk}}$, the expected number of indices $\ell$ such that $(\mathbf{e}_i^{(\ell)})^\top \mathbf{x}_i \neq 0$ is $\rho_{\mathsf{blk}}/\lambda^{\varepsilon_{\mathsf{err}}-\varepsilon_{\mathsf{LPN}}-\varepsilon_{\mathsf{wt}}}$. We call such indices *erroneous*. Thus, by an application of the Chernoff bound, we can bound the number of erroneous indices in each block. Specifically, we see that except with probability $\approx e^{-\rho_{\mathsf{blk}}/\lambda^{\varepsilon_{\mathsf{err}}-\varepsilon_{\mathsf{LPN}}-\varepsilon_{\mathsf{wt}}}}$ the number of erroneous indices in a block is bounded by $2 \cdot \rho_{\mathsf{blk}}/\lambda^{\varepsilon_{\mathsf{err}}-\varepsilon_{\mathsf{LPN}}-\varepsilon_{\mathsf{wt}}}$. Analogously, we also say that a block is *erroneous* if more than $2 \cdot \rho_{\mathsf{blk}}/\lambda^{\varepsilon_{\mathsf{err}}-\varepsilon_{\mathsf{LPN}}-\varepsilon_{\mathsf{wt}}}$ indices contained within it are erroneous.

Since our above observations were for a fixed $\mathbf{x}_i$, we now apply a union bound over all possible choices of $\mathbf{x}_i$ to argue that the probability of any fixed block $j$ being an erroneous block, for *any* $\mathbf{x}_i$ is negligible.

$$\Pr[\exists \mathbf{x}_i \text{ s.t. block } j \text{ is erroneous}] < 2^{\lambda^{\varepsilon_{\mathsf{PRG}}}} \cdot e^{-\rho_{\mathsf{blk}}/\lambda^{\varepsilon_{\mathsf{err}}-\varepsilon_{\mathsf{LPN}}-\varepsilon_{\mathsf{wt}}}}.$$

The above follows from the fact that $\mathbf{x}_i$ is generated using a PRG where the seed-size is $\lambda^{\varepsilon_{\mathsf{PRG}}}$, and to union bound over all $\mathbf{x}_i$ it is sufficient to union bound over all possible PRG seeds. Since $\varepsilon_{\mathsf{PRG}} < 1 + \varepsilon_{\mathsf{blk}} - (\varepsilon_{\mathsf{err}} - \varepsilon_{\mathsf{LPN}} - \varepsilon_{\mathsf{wt}})$, we have that the above probability is negligible.

We now want to argue that there does not exist an $\mathbf{x}_i$ such that *any* of the blocks are erroneous, except with negligible probability. Since the above argument was for a *fixed* block $j$ to be an erroneous block, we apply a union bound over the number of blocks $\rho_{\mathsf{num}} = \lambda^{\varepsilon_{\mathsf{num}}}$. Specifically, we have

$$\Pr[\exists j, \mathbf{x}_i \text{ s.t. block } j \text{ is erroneous}] < \lambda^{\varepsilon_{\mathsf{num}}} \cdot 2^{\lambda^{\varepsilon_{\mathsf{PRG}}}} \cdot e^{-\rho_{\mathsf{blk}}/\lambda^{\varepsilon_{\mathsf{err}}-\varepsilon_{\mathsf{LPN}}-\varepsilon_{\mathsf{wt}}}}.$$

Thus except with negligible probability there does not exists $\mathbf{x}_i$ or a block $j$ such that there are more than $2 \cdot \rho_{\mathsf{blk}}/\lambda^{\varepsilon_{\mathsf{err}}-\varepsilon_{\mathsf{LPN}}-\varepsilon_{\mathsf{wt}}}$ erroneous indices in each block, or alternatively the adversary can only influence at most $2 \cdot \rho_{\mathsf{blk}}/\lambda^{\varepsilon_{\mathsf{err}}-\varepsilon_{\mathsf{LPN}}-\varepsilon_{\mathsf{wt}}}$ bits in each block.

We can now rely on the blockwise resilience of $F$ to guarantee that the probability that $e_i \neq d_i$ is small inverse polynomial. In other words, with all but a small inverse polynomial probability the value $e_i$ is fixed by the hash $\mathbf{h}$.

As $\mathbf{h} \neq g^{\mathbf{0}} = 1$ is fixed and $\mathbf{s}_i^{(1)}, \ldots, \mathbf{s}_i^{(\rho)} \leftarrow \mathbb{Z}_p^\lambda$, it follows that $\mathbf{h}^{(\mathbf{s}_i^{(1)})^\top}, \ldots, \mathbf{h}^{(\mathbf{s}_i^{(\rho)})^\top}$ are uniformly distributed. Note that since $\mathbf{h}$ is fixed, these values are not under the adversary's influence. Since the output of the extractor $\mathsf{Ext}_S$ is negligibly close to uniform, we can replace $\mathsf{Ext}_S(\mathbf{h}^{(\mathbf{s}_i^{(\ell)})^\top})$ with uniform bits $u_\ell$ for each $\ell \in [\rho]$ except with negligible statistical distance. This can be done for every

Combining with our prior analysis for erroneous indices in a block, an adversary can thus effectively influence $2 \cdot \rho_{\mathsf{blk}}/\lambda^{\varepsilon_{\mathsf{err}}-\varepsilon_{\mathsf{LPN}}-\varepsilon_{\mathsf{wt}}}$ positions in any block with an adversarial choice of $\mathbf{x}_i$, with the remaining bits in the block chosen uniformly at random.

Now consider the $e_i^* = F(u_1, \ldots, u_\rho)$. Since the $u_1, \ldots, u_\rho$ are chosen uniformly random, the blockwise resilience of $F$ (Definition 4 with $q = 2 \cdot \rho_{\mathsf{blk}}/\lambda^{\varepsilon_{\mathsf{err}}-\varepsilon_{\mathsf{LPN}}-\varepsilon_{\mathsf{wt}}}$ positions under adversarial control in each block), the probability that that $F$ is constant and equal to $e_i^*$ on all vectors with Hamming distance at most $q$ from $(u_1, \ldots, u_\rho)$ is at least $1 - BI_q(F)$, where

$$BI_q(F) = O\left(\rho_{\mathsf{num}} \cdot \left(\frac{\log^2 \rho_{\mathsf{blk}}}{\rho_{\mathsf{blk}}}\right) \cdot \left(\frac{2 \cdot \rho_{\mathsf{blk}}}{\lambda^{\varepsilon_{\mathsf{err}}-\varepsilon_{\mathsf{LPN}}-\varepsilon_{\mathsf{wt}}}}\right)\right).$$

Hence, we obtain that the probability that $\mathsf{BAD}_i$ happens is at most $BI_q(F) + \mathsf{negl}(\lambda)$ (where the $\mathsf{negl}(\lambda)$ contribution is due to the randomness extractor). By the above equation the value $BI_q(F)$ is bounded by $1/\lambda^\delta$ where $\delta$ is set to be a constant smaller than $\varepsilon_{\mathsf{err}} - \varepsilon_{\mathsf{LPN}} - \varepsilon_{\mathsf{wt}} - \varepsilon_{\mathsf{num}}$. □

## Acknowledgements

The authors thank the anonymous reviewers of EUROCRYPT 2025 for their valuable feedback and Brent Waters for comments on an early draft of this work.

Pedro Branco and Giulio Malavolta are supported by the European Research Council through an ERC Starting Grant (Grant agreement No. 101077455, ObfusQation). Giulio Malavolta is also funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA – 390781972. Akshayaram Srinivasan is supported in part by a NSERC Discovery Grant RGPIN-2024-03928. Abhishek Jain is supported in part by NSF CNS-1814919, NSF CAREER 1942789, Johns Hopkins University Catalyst award, JP Morgan Faculty Award, and research gifts from Ethereum, Stellar and Cisco. Nico Döttling is funded by the European Union (ERC, LACONIC, 101041207). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

## References

[Ajt99] Miklós Ajtai. Generating hard instances of the short basis problem. In Jirí Wiedermann, Peter van Emde Boas, and Mogens Nielsen, editors, *ICALP 99: 26th International Colloquium on Automata, Languages and Programming*, volume 1644 of *Lecture Notes in Computer Science*, pages 1–9, Prague, Czech Republic, July 11–15, 1999. Springer, Berlin, Heidelberg, Germany. 21

[AL93] Miklós Ajtai and Nathan Linial. The influence of large coalitions. *Comb.*, 13(2):129–145, 1993. 12

[Ale03] Michael Alekhnovich. More on average case vs approximation complexity. In *44th Annual Symposium on Foundations of Computer Science*, pages 298–307, Cambridge, MA, USA, October 11–14, 2003. IEEE Computer Society Press. 12

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *20th Annual ACM Symposium on Theory of Computing*, pages 103–112, Chicago, IL, USA, May 2–4, 1988. ACM Press. 3

[BHHO08] Dan Boneh, Shai Halevi, Michael Hamburg, and Rafail Ostrovsky. Circular-secure encryption from decision Diffie-Hellman. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 108–125, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Berlin, Heidelberg, Germany. 16

[BKM20] Zvika Brakerski, Venkata Koppula, and Tamer Mour. NIZK from LPN and trapdoor hash via correlation intractability for approximable relations. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part III*, volume 12172 of *Lecture Notes in Computer Science*, pages 738–767, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Cham, Switzerland. 3, 4

[BL85] Michael Ben-Or and Nathan Linial. Collective coin flipping, robust voting schemes and minima of banzhaf values. In *26th Annual Symposium on Foundations of Computer Science*, pages 408–416, Portland, Oregon, October 21–23, 1985. IEEE Computer Society Press. 12

[BLMR13] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic PRFs and their applications. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 410–428, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Berlin, Heidelberg, Germany. 9

[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 Moses Charikar and Edith Cohen, editors, *51st Annual ACM Symposium on Theory of Computing*, pages 1082–1090, Phoenix, AZ, USA, June 23–26, 2019. ACM Press. 3

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

Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part I*, volume 10820 of *Lecture Notes in Computer Science*, pages 91–122, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Cham, Switzerland. 3

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004. 3

[CHK03] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 255–271, Warsaw, Poland, May 4–8, 2003. Springer, Berlin, Heidelberg, Germany. 3

[CKU20] Geoffroy Couteau, Shuichi Katsumata, and Bogdan Ursu. Non-interactive zero-knowledge in pairing-free groups from weaker assumptions. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part III*, volume 12107 of *Lecture Notes in Computer Science*, pages 442–471, Zagreb, Croatia, May 10–14, 2020. Springer, Cham, Switzerland. 3

[CZ16] Eshan Chattopadhyay and David Zuckerman. Explicit two-source extractors and resilient functions. In Daniel Wichs and Yishay Mansour, editors, *48th Annual ACM Symposium on Theory of Computing*, pages 670–683, Cambridge, MA, USA, June 18–21, 2016. ACM Press. 12

[DGI$^+$19] Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. Trapdoor hash functions and their applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 3–32, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Cham, Switzerland. 4, 5

[DJJ24] Quang Dao, Aayush Jain, and Zhengzhong Jin. Non-interactive zero-knowledge from lpn and mq. In *CRYPTO*, 2024. 3, 4

[DMP88] Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Non-interactive zero-knowledge proof systems. In Carl Pomerance, editor, *Advances in Cryptology – CRYPTO'87*, volume 293 of *Lecture Notes in Computer Science*, pages 52–72, Santa Barbara, CA, USA, August 16–20, 1988. Springer, Berlin, Heidelberg, Germany. 3

[DN00] Cynthia Dwork and Moni Naor. Zaps and their applications. In *41st Annual Symposium on Foundations of Computer Science*, pages 283–293, Redondo Beach, CA, USA, November 12–14, 2000. IEEE Computer Society Press. 12

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM Journal on Computing*, 38(1):97–139, 2008. 13

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *31st Annual Symposium on Foundations of Computer Science*, pages 308–317, St. Louis, MO, USA, October 22–24, 1990. IEEE Computer Society Press. 3, 4, 17

[FR21] Marc Fischlin and Felix Rohrbach. Single-to-multi-theorem transformations for non-interactive statistical zero-knowledge. In Juan Garay, editor, *PKC 2021: 24th International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 12711 of *Lecture Notes in Computer Science*, pages 205–234, Virtual Event, May 10–13, 2021. Springer, Cham, Switzerland. 4

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, Santa Barbara, CA, USA, August 1987. Springer, Berlin, Heidelberg, Germany. 3

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th Annual ACM Symposium on Theory of Computing*, pages 291–304, Providence, RI, USA, May 6–8, 1985. ACM Press. 3

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th Annual ACM Symposium on Theory of Computing*, pages 218–229, New York City, NY, USA, May 25–27, 1987. ACM Press. 3

[GO93] Shafi Goldwasser and Rafail Ostrovsky. Invariant signatures and non-interactive zero-knowledge proofs are equivalent (extended abstract). In Ernest F. Brickell, editor, *Advances in Cryptology – CRYPTO'92*, volume 740 of *Lecture Notes in Computer Science*, pages 228–245, Santa Barbara, CA, USA, August 16–20, 1993. Springer, Berlin, Heidelberg, Germany. 3

[GOS06a] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for NIZK. In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 97–111, Santa Barbara, CA, USA, August 20–24, 2006. Springer, Berlin, Heidelberg, Germany. 3

[GOS06b] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 339–358, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Berlin, Heidelberg, Germany. 3

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 197–206, Victoria, BC, Canada, May 17–20, 2008. ACM Press. 4, 7, 21

[HL18] Justin Holmgren and Alex Lombardi. Cryptographic hashing from strong one-way functions (or: One-way product functions and their applications). In Mikkel Thorup, editor, *59th Annual Symposium on Foundations of Computer Science*, pages 850–858, Paris, France, October 7–9, 2018. IEEE Computer Society Press. 3

[JJ21] Abhishek Jain and Zhengzhong Jin. Non-interactive zero knowledge from sub-exponential DDH. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part I*, volume 12696 of *Lecture Notes in Computer*

*Science*, pages 3–32, Zagreb, Croatia, October 17–21, 2021. Springer, Cham, Switzerland. 3, 4

[KMY20] Fuyuki Kitagawa, Takahiro Matsuda, and Takashi Yamakawa. NIZK from SNARG. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020: 18th Theory of Cryptography Conference, Part I*, volume 12550 of *Lecture Notes in Computer Science*, pages 567–595, Durham, NC, USA, November 16–19, 2020. Springer, Cham, Switzerland. 18, 19

[LPWW20] Benoît Libert, Alain Passelègue, Hoeteck Wee, and David J. Wu. New constructions of statistical NIZKs: Dual-mode DV-NIZKs and more. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part III*, volume 12107 of *Lecture Notes in Computer Science*, pages 410–441, Zagreb, Croatia, May 10–14, 2020. Springer, Cham, Switzerland. 5, 45

[Mat94] Mitsuru Matsui. Linear cryptanalysis method for DES cipher. In Tor Helleseth, editor, *Advances in Cryptology – EUROCRYPT'93*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397, Lofthus, Norway, May 23–27, 1994. Springer, Berlin, Heidelberg, Germany. 45

[Mek17] Raghu Meka. Explicit resilient functions matching ajtai-linial. In Philip N. Klein, editor, *28th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1132–1148, Barcelona, Spain, January 16–19, 2017. ACM-SIAM. 12, 45

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 700–718, Cambridge, UK, April 15–19, 2012. Springer, Berlin, Heidelberg, Germany. 14, 15, 21

[PS19] Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for NP from (plain) learning with errors. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 89–114, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Cham, Switzerland. 3, 4

[QRW19] Willy Quach, Ron D. Rothblum, and Daniel Wichs. Reusable designated-verifier NIZKs for all NP from CDH. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part II*, volume 11477 of *Lecture Notes in Computer Science*, pages 593–621, Darmstadt, Germany, May 19–23, 2019. Springer, Cham, Switzerland. 3, 17, 18, 19

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93, Baltimore, MA, USA, May 22–24, 2005. ACM Press. 13, 15

[Wat24] Brent Waters. A new approach for non-interactive zero-knowledge from learning with errors. In *STOC*, pages 399–410. ACM, 2024. 2, 4, 6, 7, 8, 9, 10, 17, 19

## A Blockwise Resilient Function

In this section, we present a simple proof for Theorem 1, where we rely on the following Theorem from [Mek17] for the resilient function underlying our construction.

**Theorem 7 ([Mek17]).** *For some universal constant $c \geq 1$ the following holds. There exists a function $f : \{0,1\}^n \mapsto \{0,1\}$ which can be computed in time $t \cdot n^c$ such that,*

- *$f$ is almost balanced:* $\Pr_{x \leftarrow_\$ \{0,1\}^n}[f(x) = 0] = 1/2 \pm 1/10.$
- *$I_q(f) \leq c \cdot q(\log^2 n)/n$.*

We now proceed to the proof of Theorem 1.

We first note the Piling-up Lemma that will be useful for the proof.

**Lemma 14 (Piling-up Lemma[Mat94]).** *Let $x_1, \ldots, x_t \in \{0,1\}$ be i.i.d. random variables such that $\mathbb{E}[x_i] = 1/2 \pm \varepsilon$, then*

$$\Pr[x_1 \oplus \ldots x_t = 0] = 1/2 \pm 2^{t-1}\varepsilon^t.$$

*Proof of Theorem 1.* We use the resilient function $f$ from Theorem 7 to construct $F$. Specifically,

$$F(x_1, \ldots, x_t) = f(x_1) \oplus \cdots \oplus f(x_t).$$

By the Piling-up Lemma, we have that

$$\Pr_{x \leftarrow_\$ \{0,1\}^{n \cdot t}}[F(x) = 0] = 1/2 \pm \frac{2^{t-1}}{10^t} = 1/2 \pm 1/2^t.$$

For the resilience, we perform a simple union bound over the $t$ blocks, which gives us $BI_q(F) \leq t \cdot I_q(f) \leq t \cdot c' \cdot q(\log^2 n)/n$. □

## B Designated-Verifier Vector Trapdoor Hash from DDH

We now present a construction in the designated-verifier model for a trapdoor hash function from DDH to show the versatility of our framework. This scheme is a recast of the scheme in [LPWW20].

$\underline{\mathsf{Setup}(1^\lambda, 1^k)}$

1. $\mathbb{G}, p, g \leftarrow \mathsf{Gen}(1^\lambda)$.
2. Sample a seed $S \leftarrow \{0,1\}^\lambda$.
3. Sample $\mathbf{a} \leftarrow \mathbb{Z}_p^n$ where $n \geq 2(\lambda + \log(p))$ and set $\mathbf{h} := g^{\mathbf{a}}$.
4. $\forall i \in [k]$,
   (a) Sample $s_i, t_i, k_i \leftarrow \mathbb{Z}_p$.
   (b) Set $\mathbf{g}_{1,i} := g^{s_i \cdot \mathbf{a}^T}$, $\mathbf{g}_{2,i} := g^{t_i \cdot \mathbf{a}^T}$ and $f_i := g^{k_i}$
   (c) Set $y_i := s_i k_i + t_i$.
   (d) Set $\mathsf{ek}_i = (S, \mathbf{g}_{1,i}, \mathbf{g}_{2,i}, f_i)$, $\mathsf{td}_i = (S, s_i)$ and $\mathsf{vk}_i = (k_i, y_i)$.
5. Set $\mathsf{hk} = (\mathbf{h}, \{\mathsf{ek}_i\}_{i \in [k]})$.

6. Output $(\mathsf{hk}, \{\mathsf{ek}_i, \mathsf{td}_i, \mathsf{vk}_i\}_{i\in[k]})$.

$\underline{\mathsf{Hash}(\mathsf{hk}, \mathbf{x} \in \{0,1\}^n)}$

1. Parse $\mathsf{hk} = (\mathbf{h}, \{\mathsf{ek}_i\}_{i\in[k]})$ and $\mathsf{ek}_i = (S, \mathbf{g}_{1,i}, \mathbf{g}_{2,i}, f_i)$.
2. Compute $h \coloneqq \mathbf{h}^{\mathbf{x}} (= g^{\mathbf{a}^T \cdot \mathbf{x}})$.
3. $\forall i \in [k]$,
   (a) Set $\gamma_i \coloneqq \mathbf{g}_{1,i}^{\mathbf{x}} (= g^{s_i \cdot (\mathbf{a}^T \cdot \mathbf{x}^T)})$ and $\delta_i \coloneqq \mathbf{g}_{2,i}^{\mathbf{x}} (= g^{t_i \cdot (\mathbf{a}^T \cdot \mathbf{x})})$.
   (b) Set $\boldsymbol{\pi}_i \coloneqq (\gamma_i, \delta_i)$.
4. Output $(h, \{\boldsymbol{\pi}_i\}_{i\in[k]})$.

$\underline{\mathsf{Encode}(\mathsf{ek}_i, \pi_i)}$

1. Parse $\mathsf{ek}_i = (S, \mathbf{g}_{1,i}, \mathbf{g}_{2,i}, f_i)$ and $\boldsymbol{\pi}_i = (\gamma_i, \delta_i)$.
2. Output $e_i = \mathsf{Ext}_S(\gamma_i)$. //$\mathsf{Ext}_S$ outputs the least significant bit.

$\underline{\mathsf{Decode}(\mathsf{td}_i, h)}$

1. Parse $\mathsf{td}_i = (S, s_i)$.
2. Output $d_i = \mathsf{Ext}_S(h^{s_i})$.

$\underline{\mathsf{Verify}(\mathsf{hk}, \mathsf{vk}_i, h, i, \pi_i, \widetilde{e})}$

1. Parse $\mathsf{hk} = \mathbf{h}$, $\mathsf{vk}_i = (k_i, y_i)$, and $\boldsymbol{\pi}_i = (\gamma_i, \delta_i)$.
2. Check if $h^{y_i} = \gamma_i^{k_i} \delta_i$ and if $\widetilde{e} = \mathsf{Ext}_S(\gamma_i)$.
3. Output 1 if all the checks verify.

**Lemma 15 (Completeness and succinctness).** *The scheme presented above is complete and succinct.*

Succinctness follows immediately since $h = g^{\mathbf{a}^T \cdot \mathbf{x}}$ is composed by only one group element. Hence, $|h| = \log(p) = \mathsf{poly}(\lambda)$. Moreover, completeness follows since $h^{y_i} = g^{(\mathbf{a}^T \cdot \mathbf{x}) s_i k_i} g^{(\mathbf{a}^T \cdot \mathbf{x}) t_i} = \gamma_i^{k_i} \delta_i$.

## B.1 Hiding

We prove the hiding property of this construction below.

**Lemma 16.** *The scheme presented above is secure assuming that the DDH assumption is hard.*

*Proof.* We will start by proving mode indistinguishability.

**Mode indistinguishability.** We will first show how the $\mathsf{Setup}^*$ algorithm samples the encoding keys. We will replace each one at a time.

Let $i^* \in [k]$ be any index. The proof follows the following sequence of hybrids.

$\mathsf{Hyb}_0$**:** This hybrid generates the distribution as in the 'real' game. Specifically,

1. Sample a seed $S \leftarrow \{0,1\}\lambda$.
2. Sample $\mathbf{a} \leftarrow \mathbb{Z}_p$ and set $\mathbf{h} := g^{\mathbf{a}}$. Set $\mathsf{hk} = \mathbf{h}$.
3. $\forall i \in [k]$,
   (a) Set $\mathbf{g}_{1,i} := g^{s_i \cdot \mathbf{a}^T}$, $\mathbf{g}_{2,i} := g^{t_i \cdot \mathbf{a}^T}$ and $f_i := g^{k_i}$
   (b) Set $y_i := s_i k_i + t_i$.
   (c) Set $\mathsf{ek}_i = (S, \mathbf{g}_{1,i}, \mathbf{g}_{2,i}, f_i)$, $\mathsf{td}_i = (S, s_i)$ and $\mathsf{vk}_i = (k_i, y_i)$.

$\mathsf{Hyb}_1$**:** In this hybrid, we sample $k_{i^*}, y_{i^*}$ randomly and set $t_{i^*} = y_{i^*} - s_{i^*} k_{i^*}$. Specifically,

1. Sample a seed $S \leftarrow \{0,1\}\lambda$.
2. Sample $\mathbf{a} \leftarrow \mathbb{Z}_p$ and set $\mathbf{h} := g^{\mathbf{a}}$. Set $\mathsf{hk} = \mathbf{h}$.
3. $\forall i \neq i^*$,
   (a) Set $\mathbf{g}_{1,i} := g^{s_i \cdot \mathbf{a}^T}$, $\mathbf{g}_{2,i} := g^{t_i \cdot \mathbf{a}^T}$ and $f_i := g^{k_i}$
   (b) Set $y_i := s_i k_i + t_i$.
   (c) Set $\mathsf{ek}_i = (S, \mathbf{g}_{1,i}, \mathbf{g}_{2,i}, f_i)$, $\mathsf{td}_i = (S, s_i)$ and $\mathsf{vk}_i = (k_i, y_i)$.
4. $\underline{\text{Sample } k_{i^*}, y_{i^*} \leftarrow \mathbb{Z}_p \text{ and set } t_{i^*} = y_{i^*} - s_{i^*} k_{i^*}.}$
5. $\underline{\text{Set } \mathbf{g}_{1,i^*} := g^{s_{i^*} \cdot \mathbf{a}^T}, \mathbf{g}_{2,i^*} := g^{t_{i^*} \cdot \mathbf{a}^T} \text{ and } f_{i^*} := g^{k_{i^*}}.}$
6. $\underline{\text{Set } \mathsf{ek}_{i^*} = (S, \mathbf{g}_{1,i^*}, \mathbf{g}_{2,i^*}, f_{i^*}), \mathsf{td}_{i^*} = (S, s_{i^*}) \text{ and } \mathsf{vk}_{i^*} = (k_{i^*}, y_{i^*}).}$

$\underline{\mathsf{Hyb}_1 \approx_s \mathsf{Hyb}_0.}$ . Perfect indistinguishability of hybrids follows from the fact that $t_{i^*} = y_{i^*} - s_{i^*} k_{i^*}$.

$\mathsf{Hyb}_2$**:** In this hybrid we set $\mathbf{g}_{1,i}$ as $g^{\mathbf{v}}$ where $\mathbf{v} \leftarrow\!\!\$\ \mathbb{Z}_p^n$.

1. Sample a seed $S \leftarrow \{0,1\}\lambda$.
2. Sample $\mathbf{a} \leftarrow \mathbb{Z}_p$ and set $\mathbf{h} := g^{\mathbf{a}}$. Set $\mathsf{hk} = \mathbf{h}$.
3. $\forall i \neq i^*$,
   (a) Set $\mathbf{g}_{1,i} := g^{s_i \cdot \mathbf{a}^T}$, $\mathbf{g}_{2,i} := g^{t_i \cdot \mathbf{a}^T}$ and $f_i := g^{k_i}$
   (b) Set $y_i := s_i k_i + t_i$.
   (c) Set $\mathsf{ek}_i = (S, \mathbf{g}_{1,i}, \mathbf{g}_{2,i}, f_i)$, $\mathsf{td}_i = (S, s_i)$ and $\mathsf{vk}_i = (k_i, y_i)$.
4. Sample $k_{i^*}, y_{i^*} \leftarrow \mathbb{Z}_p$.
5. $\underline{\text{Set } \mathbf{g}_{1,i^*} := g^{\mathbf{v}^T} \text{ for } \mathbf{v} \leftarrow\!\!\$\ \mathbb{Z}_p^n}$, $\mathbf{g}_{2,i^*} := g^{(y_{i^*} - s_{i^*} k_{i^*}) \cdot \mathbf{a}^T}$ and $f_{i^*} := g^{k_{i^*}}$.
6. Set $\mathsf{ek}_{i^*} = (S, \mathbf{g}_{1,i^*}, \mathbf{g}_{2,i^*}, f_{i^*})$, $\mathsf{td}_{i^*} = (S, s_{i^*})$ and $\mathsf{vk}_{i^*} = (k_{i^*}, y_{i^*})$.

$\underline{\mathsf{Hyb}_2 \approx_c \mathsf{Hyb}_1.}$ Indistinguishability of hybrids follows from the DDH assumption which states that

$$(g^{\mathbf{a}}, g^{s_{i^*}}, g^{s_{i^*}\mathbf{a}}) \approx_c (g^{\mathbf{a}}, g^{s_{i^*}}, g^{\mathbf{v}})$$

where $\mathbf{v} \leftarrow\!\!\$\ \mathbb{Z}_p^n$. Additionally, note that given $g^{s_{i^*}\mathbf{a}}$ (or $g^{\mathbf{v}}$) the reduction can efficiently simulate $\mathsf{vk}_{i^*} = (k_{i^*}, y_{i^*})$ where $k_{i^*}, y_{i^*} \leftarrow \mathbb{Z}_p$, and $\mathbf{g}_{2,i^*} = g^{(y_{i^*} - k_{i^*} s_{i^*})\mathbf{a}}$.

We set the $\mathsf{ek}_i$ in the last hybrid to be the output of $\mathsf{Setup}^*$

**Pseudorandomness.** Having set the $\mathsf{ek}_{i^*}$ to be to be the output of $\mathsf{Setup}^*$ we argue that the bit $e_{i^*}$ is indistinguishable from uniform.

Indistinguishability follows from the leftover hash lemma (Lemma 3). To see this note that $\mathbf{g}_{1,i^*} = g^{\mathbf{v}}$ and that $e_{i^*}$ is computed as $\mathsf{Ext}_S(\gamma_{i^*})$ where $\gamma_{i^*} = \mathbf{g}_{1,i^*}^{\mathbf{x}} = g^{\mathbf{v}\cdot\mathbf{x}^T}$. Hence, it is enough to show that $\gamma_{i^*}$ is indistinguishable from uniform.

First note that $\tilde{H}_\infty(\mathbf{x}) = n$ and that $|\mathbf{a}^T\mathbf{x}| = \log p$. Hence we can use a chain rule (Lemma 2)

$$\tilde{H}_\infty(\mathbf{x}|\mathbf{a}^T\mathbf{x}) \geq \tilde{H}_\infty(\mathbf{x}) - \log p.$$

We can now use the leftover hash lemma to argue that

$$(\mathbf{v}^T\mathbf{x}, \mathbf{a}^T\mathbf{x}) \approx_s (u, \mathbf{a}^T\mathbf{x})$$

where $u \leftarrow \mathbb{Z}_p$ as long as $\log p \leq n - \log p - 2\lambda$. □

## B.2 Statistical Binding

We prove the binding property of this construction below.

**Lemma 17.** *The scheme presented above is statistically binding.*

*Proof.* Assume that for an adversarially chosen hash value $h$, index $i$ and proof $\pi$, we have that $\mathsf{Encode}(\mathsf{ek}_i, \pi_i) = e_i \neq d_i = \mathsf{Decode}(\mathsf{td}_i, h)$. This means that $h^{s_i} \neq \gamma_i$. Let $h = g^w, \gamma_i = g^{z_0}$ and $\delta_i = g^{z_1}$ for some $w, z_0, z_1 \in \mathbb{Z}_p$. Then

$$\begin{aligned}
\Pr_{k_i \leftarrow \mathbb{Z}_p}[\mathsf{Verify}(\mathsf{hk}, \mathsf{vk}_i, h, i, \pi_i, \widetilde{e}) = 1 : h^{s_i} \neq \gamma_i] &= \Pr_{k_i \leftarrow \mathbb{Z}_p}\left[h^{s_i \cdot k_i + t_i} = \gamma_i^{k_i}\delta_i : h^{s_i} \neq \gamma_i\right] \\
&= \Pr_{k_i \leftarrow \mathbb{Z}_p}[w(s_i \cdot k_i + t_i) = z_0 k_i + z_1 : ws_i \neq z_0] \\
&= \Pr_{k_i \leftarrow \mathbb{Z}_p}\left[k_i = \frac{w(s_i + t_i) - z_1}{z_0 - ws_i} : ws_i \neq z_0\right] \\
&= \frac{1}{p} = \frac{1}{2^\lambda} = \mathsf{negl}(\lambda).
\end{aligned}$$

We conclude that if $e_i \neq d_i$, then $\mathsf{Verify}(\mathsf{hk}, \mathsf{vk}_i, h, i, \pi_i, \widetilde{e})$ fails except with negligible probability, even when the adversary is computationally unbounded. □