CSC 2419 Project Report

Methods of achieving Non-Interactive Zero-Knowledge using Learning with Errors

William He

Instructor: Akshayaram Srinivasan

December 11, 2025

Abstract

We will summarise the method of constructing non-interactive zero-knowledge (NIZK) proof systems using the hidden bits model assuming Learning with Errors (LWE). We will first look at such construction by Waters (STOC' 24), then we will talk about the improvement to the explanation of the existing construction by Branco et al. (EUROCRYPT 2025), to reach a much more simple proof. The construction involves a LWE-based implementation of a hidden bits generator that is very different to the traditional method of using correlation intractability. The construction uses simple LWE assumptions and lattice trapdoors.

1 Introduction

Zero-knowledge proof (ZKP) [GMR85] allows a prover to prove the validity of a statement to a verifier without the verifier learning why the statement is true. This is done by a sequence of interactions between the prover and the verifier. It is well known that ZKPs for the class NP can be obtained by constructing a ZKP for an NP-complete language.

For efficiency, the notion of non-interactive zero-knowledge proofs (NIZKs) was introduced to reduce communication overhead by replacing the interaction with a single message. NIZKs for NP can be obtained by using a hash function via the Fiat–Shamir heuristic [FS86], which is usually modeled as a random oracle. Instead, we require the hash function to satisfy the correlation intractability property [CGH04].

In this report, we study a completely different approach called the hidden-bit generator by Waters [Wat24], based on the Learning with Errors (LWE) assumption. This approach does not rely on sophisticated operations beyond lattice trapdoors. We also discuss how some parts of the original explanation can be simplified, following Branco et al. [BCD+25].

2 Prerequisites

2.1 Hidden Bits Generators

We first define a hidden-bit generator.

Definition 2.1 (Hidden-Bit Generator)**.** *A hidden-bit generator Π_{HBG} is a set of probabilistic polynomial-time (PPT) algorithms*

- $\mathsf{Setup}(1^\lambda, 1^k) \to \mathsf{crs}$*: On input the security parameter λ and the length parameter k, the algorithm outputs a common reference string* crs*.*
- $\mathsf{GenBits}(\mathsf{crs}) \to (\mathsf{com}, \mathbf{r}, (\pi_1, \ldots, \pi_k))$*: On input the common reference string* crs*, the algorithm outputs a k-bit string $\mathbf{r} = (r_1, \ldots, r_k) \in \{0,1\}^k$, its commitment* com*, and k openings $(\pi_1, \ldots, \pi_k)$, where each π_i opens the i^{th} index of the commitment.*

- $\mathsf{Verify}(\mathsf{crs}, \mathsf{com}, i, \beta, \pi) \to b$: *On input the common reference string* crs*, the index* i*, a bit* $\beta \in \{0,1\}$*, and an opening* π*, the algorithm outputs a bit* $b \in \{0,1\}$*. The algorithm* Verify *outputs* 1 *if* β *is the opened value of the* i^{th} *index of* com *using the opening* π*.*

The above three algorithms must satisfy the following properties.

- ***Correctness:*** *For all* $\lambda, k \in \mathbb{N}$*,*

$$\Pr\left[\mathsf{Verify}(\mathsf{crs}, \mathsf{com}, i, r_i, \pi_i) = 1 : \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^k) \\ (\mathsf{com}, \mathbf{r}, (\pi_1, \ldots, \pi_k)) \leftarrow \mathsf{GenBits}(\mathsf{crs}) \end{array}\right] = 1.$$

- ***Statistical Binding:*** *For every* crs *in the support of* $\mathsf{Setup}(1^\lambda, 1^k)$*, there exists a set* $\mathcal{V}^{\mathsf{crs}} \subseteq \{0,1\}^k$ *that satisfies the following two properties.*

  1. ***Output sparsity:*** *There exists a constant* $v < 1$ *and a polynomial* $p(\cdot)$ *such that for every polynomial* $k(\lambda)$*, there exists* λ_0 *such that for all* $\lambda > \lambda_0$ *and for every* crs *in the support of* $\mathsf{Setup}(1^\lambda, 1^k)$*, where* $k = k(\lambda)$*, we have* $|\mathcal{V}^{\mathsf{crs}}| \leq 2^{k^v \cdot p(\lambda)}$*.*
  2. ***Statistical binding game:*** *Consider the following game between a challenger and a computationally unbounded adversary* $\mathcal{A}$*.*
     (a) *On input the security parameter* 1^λ*,* $\mathcal{A}$ *outputs* 1^k*.*
     (b) *The challenger samples* $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^k)$ *and sends* crs *to the adversary* $\mathcal{A}$*.*
     (c) *The adversary* $\mathcal{A}$ *outputs* $(\mathsf{com}, I \subseteq [k], \mathbf{r}_I, \{\pi_i\}_{i \in I})$*.*
     (d) *The experiment outputs* 1 *if* $\mathsf{Verify}(\mathsf{crs}, \mathsf{com}, i, r_i, \pi_i) = 1$ *for all* $i \in I$ *and* $\mathbf{r}_I \notin \mathcal{V}_I^{\mathsf{crs}}$*. Otherwise, the experiment outputs* 0*.*

     We say Π_{HBG} *is statistical binding if for all adversaries* $\mathcal{A}$*,* $\Pr[b = 1] \leq negl(\lambda)$*.*

- ***Single-Bit Hiding:*** *Let* $b \in \{0,1\}$ *be a uniformly random bit. Consider the following game between a challenger and a PPT adversary* $\mathcal{A}$*.*

  1. *On input the security parameter* 1^λ*,* $\mathcal{A}$ *outputs* 1^k *and a query index* $i^* \in [k]$*.*
  2. *The challenger samples* $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^k)$ *and then computes* $(\mathsf{com}, \mathbf{r}, (\pi_1, \ldots, \pi_k)) \leftarrow \mathsf{GenBits}(\mathsf{crs})$*.*
  3. *The challenger sends* crs*,* com*, and* (r_i, π_i) *for all* $i \neq i^*$ *to the adversary* $\mathcal{A}$*.*
  4. *If* $b = 0$*, the challenger sends* r_{i^*} *to the adversary. If* $b = 1$*, the challenger sends a random bit* $u \in \{0,1\}$ *to the adversary.*
  5. *The adversary* $\mathcal{A}$ *outputs a bit* b'*, where it outputs* 0 *if it believes the bit is the true* r_{i^*} *and* 1 *if it believes the bit is random.*

  We say Π_{HBG} *is single-bit hiding if for all PPT adversaries* $\mathcal{A}$*,*

$$|\Pr[b' = 1 \mid b = 1] - \Pr[b' = 1 \mid b = 0]| \leq negl(\lambda).$$

The original definition of the hiding property is called *adaptive computational hiding*. This is different from the hiding property defined above. By previous works [FLS90, KMY20, QRW19], we have the following theorem.

Theorem 2.2 (NIZK from Hidden-Bit Generator, Original Definition)**.** *If there exists a hidden-bit generator as in Definition 2.1, except that the hiding property is replaced by adaptive computational hiding, then there exists a computational NIZK for* NP*.*

It is proven in [Wat24] that if a hidden-bit generator satisfies single-bit hiding, then it also satisfies adaptive computational hiding. Therefore, we obtain the following result.

Theorem 2.3 (NIZK from Hidden-Bit Generator, Definition 2.1)**.** *If there exists a hidden-bit generator according to Definition 2.1, then there exists a computational NIZK for* NP*.*

Therefore, it suffices to construct a hidden-bit generator that satisfies Definition 2.1.

2.2 Results from LWE and Lattice Trapdoors

We will summarise results in LWE and lattice trapdoors that will be useful, and they applies according to our settings. We will assume all norms are l^∞ norm.

Definition 2.4 (Bounded Discrete Gaussians)**.** *Let $\sigma = \sigma(\lambda)$ be the Gaussian width parameter.*

We denote D_σ to represent a discrete Gaussian distribution over $\mathbb{Z}$ with standard deviation $\sigma \in \mathbb{R}^+$.

We also denote $\tilde{D}_\sigma$ to be the same as D_σ except it is 0 whenever the norm exceeds $\sqrt{\lambda}\sigma$.

Proposition 2.5.

$$\tilde{D}_\sigma \approx_s D_\sigma$$

Claim 2.6 (Learning with Errors)**.** *Given λ be a security parameter and let n, m, q be integers, σ be a Gaussian width parameter. Then for $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}, \mathbf{s} \leftarrow \mathbb{Z}_q^n, \mathbf{e} \leftarrow D_\sigma^m, \mathbf{u} \leftarrow \mathbb{Z}_q^m$*

$$(\mathbf{A}, \mathbf{s}^\top\mathbf{A} + \mathbf{e}^\top) \approx_c (\mathbf{A}, \mathbf{u})$$

Theorem 2.7 (Lattice Trapdoors [GPV08])**.** *Given parameters n, m, q, the following two PPT algorithms exists*

- $\mathsf{TrapGen}(1^n) \to (\mathbf{A}, \mathsf{td}_A)$: *On input n, m, q, the algorithm outputs a matrix $A \in \mathbb{Z}_q^{n\times m}$ and its trapdoor td_A.*
- $\mathsf{TrapPreSamp}(\mathbf{A}, \mathsf{td}_A, \mathbf{v})$: *On input a matrix $\mathbf{A}$, a trapdoor td_A, a target vector $\mathbf{v}$. The algorithm outputs a vector $\mathbf{u}$.*

where the matrix A output by $\mathsf{TrapGen}$ is statistically closed to uniform. Consider $(\mathbf{A}, \mathsf{td}) \leftarrow \mathsf{TrapGen}(1^n)$, we also have

$$(\mathbf{A}, \mathbf{e} \leftarrow \mathsf{TrapPreSamp}(\mathbf{A}, \mathsf{td}, \mathbf{u}), \mathbf{u} \leftarrow \mathbb{Z}_q^n) \approx_s (\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}, \mathbf{e} \leftarrow \tilde{D}_\sigma^m, \mathbf{u} = A \cdot e)$$

3 Hidden Bit Generators Construction

Given λ, k, we will consider the following parameters' value selection, pick any constant $\gamma \in (0, 0.5)$, take

- $q : 2^\lambda < q < 2^{\lambda+1}$ and q is prime
- $n : \lambda^{1/\gamma}$
- $\sigma = n^{1.1}$
- $m = 2(n+1)\log(q)$
- $L = \lambda \cdot m \cdot k + 5\lambda + 1$
- $B = 2^{0.5\lambda}$
- $\mathsf{TestBound} = \sigma\sqrt{\lambda}LB$
- $\mathsf{RoundBound} = \sigma\sqrt{\lambda}m\mathsf{TestBound}$

Note that n, σ, m, L are in $poly(\lambda)$. Moreover, the results from LWE and lattice trapdoors applies in our settings. The algorithms works as follows,

Algorithm 3.1.

- $\mathsf{Setup}(1^\lambda, 1^k)$:
  1. *Sample $(\mathbf{A}_i, \mathsf{td}_i) \leftarrow \mathsf{TrapGen}(1^n)$ for all $i \in [k]$*
  2. *Sample random $\mathbf{U} \in \mathbb{Z}_q^{n\times L}$*
  3. *For all $i \in [k]$, sample*
     (a) $\mathbf{s}_i \in \mathbb{Z}_q^n$
     (b) $\mathbf{e}_i \in \tilde{D}_\sigma^m$
     (c) $d_i \in \mathbb{Z}_q$

4. *Compute* $\mathbf{v}_i = \mathbf{s}_i^\top \mathbf{A}_i + \mathbf{e}_i^\top$ *for all* $i \in [k]$
5. *Sample* $\mathbf{W}_i \leftarrow \mathsf{TrapPreSamp}(\mathbf{A}_i, \mathsf{td}_i, \mathbf{U})$ *for all* $i \in [k]$
6. *Output* $\mathsf{crs} = \{\mathbf{U}, \mathbf{A}_i, \mathbf{W}_i, \mathbf{v}_i, d_i\}_{i\in[k]}$

- $\mathsf{GenBits}(\mathsf{crs})$:

1. *Sample random* $\mathbf{t} \in [-B, B]^L$
2. *Compute* $\pi_i = \mathbf{W}_i\mathbf{t}$ *for all* $i \in [k]$
3. *Set* $r_i = \mathsf{round}(\mathbf{v}_i^\top \pi_i + d_i)$ *for all* $i \in [k]$
4. *Compute* $\mathsf{com} = \mathbf{Ut}$
5. *Output* $(\mathsf{com}, \mathbf{r}, (\pi_1, \ldots, \pi_k))$

- $\mathsf{Verify}(\mathsf{crs}, \mathsf{com}, i, \beta, \pi)$:

1. *Check if* $\|\pi\| \leq \mathsf{TestBound}$
2. *Check if* $\mathsf{com} = \mathbf{A}_i\pi$
3. *Check if* $\beta = \mathsf{round}(\mathbf{v}_i^\top \pi + d_i)$
4. *Check if* $\beta = \mathsf{round}(\mathbf{v}_i^\top \pi + d_i + \mathsf{RoundBound})$ *and* $\beta = \mathsf{round}(\mathbf{v}_i^\top \pi + d_i - \mathsf{RoundBound})$
5. *Output* 1 *if all above tests hold and* 0 *otherwise.*

Note: $\mathsf{round} : \mathbb{Z}_q \to \{0, 1\}$ *is the spooky rounding function*

4 Proof of Correctness

We first show that this system has negligible correctness error.

Theorem 4.1. *Algorithm 3.1 satisfies the correctness of Definition 2.1 except for negligible probability*

Proof. Suppose $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^k), (\mathsf{com}, \mathbf{r}, (\pi_1, \ldots, \pi_k)) \leftarrow \mathsf{GenBits}(\mathsf{crs})$.

For the first test, under the correct opening π for index i we should have $\pi = \pi_i$. $\mathbf{W}_i$ obtained from $\mathsf{TrapPreSamp}$ is statistically close to $\tilde{D}_\sigma^{m\times L}$. Hence, $\mathbf{W}_i$ have entries of at most $\sigma\sqrt{\lambda}$. Since we also have $\mathbf{t} \in [-B, B]^L$

$$\|\pi_i\| = \|\mathbf{W}_i\mathbf{t}\| \leq \sigma\sqrt{\lambda}LB = \mathsf{TestBound}$$

For the second test, under the correct opening π for index i we should have $\pi = \pi_i$, hence

$$\begin{aligned}\mathbf{A}_i\pi_i &= \mathbf{A}_i\mathbf{W}_i\mathbf{t} \quad [\text{How } \pi_i \text{ is computed in } \mathsf{GenBits}]\\ &= \mathbf{Ut} \quad [\mathbf{W}_i \text{ is obtained by } \mathsf{TrapPreSamp}(\mathbf{A}_i, \mathsf{td}_i, \mathbf{U})]\\ &= \mathsf{com}\end{aligned}$$

so this check must pass.

For the third test, under the correct bit β for index i we should have $\beta = r_i$. The check $\beta = \mathsf{round}(\mathbf{v}_i^\top \pi + d_i)$ must pass as this is how r_i is defined in $\mathsf{GenBits}$.

Finally, the last check uses spooky rounding technique to check if the rounded output would flip when adding or subtracting $\mathsf{RoundBound}$ to $\mathbf{v}_i^\top \pi + d_i$. This fails with a probability

$$\frac{4\mathsf{RoundBound}}{q} = \frac{4\sigma^2\lambda LB}{q} = \frac{2^{0.5\lambda}poly(\lambda)}{q} \leq \frac{poly(\lambda)}{2^{0.5\lambda}} \leq negl(\lambda)$$

□

Now, we need to obtain perfect correctness, consider the following $\mathsf{NewSetup}, \mathsf{NewVerify}$ algorithm

Algorithm 4.2.

- $\mathsf{NewGenBits}(\mathsf{crs})$:

1. *Run the original* $\mathsf{GenBits}$ *algorithm step 1-4*

2. *For all* $i \in [k]$
   (a) *Run* $\mathsf{Verify}(\mathsf{crs}, \mathsf{com}, i, r_i, \pi_i)$
   (b) *If* Verify *rejects, set* $\mathsf{com} = \perp, \mathbf{r} = 0^k$
3. *Output* $(\mathsf{com}, \mathbf{r}, (\pi_1, \ldots, \pi_k))$

- $\mathsf{NewVerify}(\mathsf{crs}, \mathsf{com}, i, \beta, \pi)$:

1. *If* $\mathsf{com} = \perp$:
   (a) *Output* 1 *if* $\beta = 0$ *and* 0 *otherwise.*
2. *Else:*
   (a) *Check if* $\|\pi\| \leq \mathsf{TestBound}$
   (b) *Check if* $\mathsf{com} = A_i\pi$
   (c) *Check if* $\beta = \mathsf{round}(\mathbf{v}_i^\top \pi + d_i)$
   (d) *Check if* $\beta = \mathsf{round}(\mathbf{v}_i^\top \pi + d_i + \mathsf{RoundBound})$ *and* $\beta = \mathsf{round}(\mathbf{v}_i^\top \pi + d_i - \mathsf{RoundBound})$
   (e) *Output* 1 *if all above tests hold and* 0 *otherwise.*

Theorem 4.3. *If the system in Algorithm 3.1 is statistical binding and single-bit hiding then the new system that replaces* GenBits, Verify *algorithms to Algorithm 4.2 satisfies all the properties in Definition 2.1*

Proof. The new system using Algorithm 4.2 is obviously perfectly correct, as NewGenBits tests if there is any correctness error before outputing the result, and NewVerify is now adapted so that it will consider the case if there is any output error in NewGenBits.

If the original system is statistical binding, in the new system $\mathsf{com} = \perp$ only opens to 0^k, so we cannot let the adversary to output any substring of 0^k if the value of the commitment is $\perp$, to do this we need to include 0^k in $\mathcal{V}^{\mathsf{crs}}$. This achieves statistical binding. Adding one string to the set does not affect the output sparsity property.

If the original system is single-bit hiding, the new system adds a negligible probability of an adversary $\mathcal{A}$ winning the single-bit hiding game. This negligible chance occurs when $\mathsf{com} = \perp$, in this case, $\mathcal{A}$ can just guess 0. However, since the probability of $\mathsf{com} = \perp$ is negligible, the overall probability of $\mathcal{A}$ winning the game is still negligible. □

This shows that it is enough for us to show binding and hiding property of the original algorithm.

5 Proof of Binding Security

Theorem 5.1. *Algorithm 3.1 achieves statistical binding of Definition 2.1 where* $|\mathcal{V}^{\mathsf{crs}}| = 2^{(\lambda+1)n}$

Proof. First note that the value $2^{(\lambda+1)n}$ does achieve output sparsity as it is in $2^{k^v \cdot p(\lambda)}$ where $v = 0$

Now, given each crs, since $\mathsf{com} \in \mathbb{Z}_q^n$. This gives us at most $q^n \leq 2^{(\lambda+1)n}$ commitments. We will now show that each commitment com, it opens to at most one bit for each index i.

If Verify accepts, then we must have $\mathsf{com} = \mathbf{A}_i\pi$. Therefore

$$\beta = \mathsf{round}(\mathbf{v}_i^\top \pi + d_i) = \mathsf{round}((\mathbf{s}_i^\top \mathbf{A}_i + \mathbf{e}_i^\top)\pi + d_i) = \mathsf{round}(\mathbf{s}_i^\top \mathsf{com} + d_i + \mathbf{e}_i^\top \pi) \tag{1}$$

We show this only opens to $\mathsf{round}(\mathbf{s}_i^\top \mathsf{com} + d_i)$. Since $\mathbf{e}_i^\top \in \tilde{D}_\sigma^m, \|\pi\| \leq \mathsf{TestBound}$, we have

$$\|\mathbf{e}_i^\top \pi\| \leq \sqrt{\lambda}\sigma m \mathsf{TestBound} = \mathsf{RoundBound}$$

Therefore, if we have $\mathsf{round}(\mathbf{s}_i^\top \mathsf{com} + d_i + \mathbf{e}_i^\top \pi) \neq \mathsf{round}(\mathbf{s}_i^\top \mathsf{com} + d_i)$ then one of the test on step 4 will be evaluated to $\mathsf{round}(\mathbf{s}_i^\top \mathsf{com} + d_i)$ and not $\mathsf{round}(\mathbf{s}_i^\top \mathsf{com} + d_i + \mathbf{e}_i^\top \pi) = \beta$, hence Verify will reject.

Now, since each commitment opens to at most one bit for each index, each commitment opens to at most one string. Hence, $\mathcal{V}^{\mathsf{crs}}$ can include those $q^n \leq 2^{(\lambda+1)n}$ strings to achieve statistical binding property. □

Remark 5.2. *Algorithm 3.1 reaches perfect binding as if adversary obtains the correct opening values for index* i *for all* $i \in I$*, where* I *is some index set, then it must be the case that the adversary output* $\mathbf{r}_I \in \mathcal{V}_I^{\mathsf{crs}}$ *as each commitment opens to at most one string and each index opens to at most one bit.*

6 Proof of Hiding Security

We will provide an alternate proof by Branco et al. [BCD+25]. We define a sequence of different setup algorithms which is computationally indistinguishable to each other. Consider the following new setup algorithms, for a fixed i^*

Algorithm 6.1. $\mathsf{NewSetup}_{i^*}(1^\lambda, 1^k)$:

1. *Sample* $\mathbf{A}_{i^*} \in \mathbb{Z}_q^{n\times m}, \mathbf{W}_{i^*} \in D_\sigma^{m\times L}$
2. $\mathbf{U} = \mathbf{A}_{i^*}\mathbf{W}_{i^*}$
3. *Sample* $(\mathbf{A}_i, \mathsf{td}_i) \leftarrow \mathsf{TrapGen}(1^n)$ *for all* $i \in [k]\backslash\{i^*\}$
4. *Sample, for all* $i \in [k]$
   (a) $\mathbf{s}_i \in \mathbb{Z}_q^n$
   (b) $\mathbf{e}_i \in \tilde{D}_\sigma^m$
   (c) $d_i \in \mathbb{Z}_q$
5. *Compute* $\mathbf{v}_i = \mathbf{s}_i^\top \mathbf{A}_i + \mathbf{e}_i^\top$ *for all* $i \in [k]$
6. *Sample* $\mathbf{W}_i \leftarrow \mathsf{TrapPreSamp}(\mathbf{A}_i, \mathsf{td}_i, \mathbf{U})$ *for all* $i \in [k]\backslash\{i^*\}$
7. *Output* $\mathsf{crs} = \{\mathbf{U}, \mathbf{A}_i, \mathbf{W}_i, \mathbf{v}_i, d_i\}_{i\in[k]}$

Algorithm 6.2. $\mathsf{NewNewSetup}_{i^*}(1^\lambda, 1^k)$:

1. *Sample* $\mathbf{A}_{i^*} \in \mathbb{Z}_q^{n\times m}, \mathbf{W}_{i^*} \in \tilde{D}_\sigma^{m\times L}$
2. $\mathbf{U} = \mathbf{A}_{i^*}\mathbf{W}_{i^*}$
3. *Sample* $(\mathbf{A}_i, \mathsf{td}_i) \leftarrow \mathsf{TrapGen}(1^n)$ *for all* $i \in [k]\backslash\{i^*\}$
4. *Sample, for all* $i \in [k]\backslash\{i^*\}$
   (a) $\mathbf{s}_i \in \mathbb{Z}_q^n$
   (b) $\mathbf{e}_i \in \tilde{D}_\sigma^m$
   (c) $d_i \in \mathbb{Z}_q$
5. *Compute* $\mathbf{v}_i = \mathbf{s}_i^\top \mathbf{A}_i + \mathbf{e}_i^\top$ *for all* $i \in [k]\backslash\{i^*\}$.
6. *Sample* $\mathbf{v}_{i^*} \leftarrow \mathbb{Z}_q^m$
7. *Sample* $\mathbf{W}_i \leftarrow \mathsf{TrapPreSamp}(\mathbf{A}_i, \mathsf{td}_i, \mathbf{U})$ *for all* $i \in [k]\backslash\{i^*\}$
8. *Output* $\mathsf{crs} = \{\mathbf{U}, \mathbf{A}_i, \mathbf{W}_i, \mathbf{v}_i, d_i\}_{i\in[k]}$

We now state a sequence of propositions which state that the system with different . We provide the proofs in the Appendix

Proposition 6.3. *By the trapdoor pre-sampling property, the output from* Setup *in Algorithm 3.1 and the output from* $\mathsf{NewSetup}_{i^*}$ *in Algorithm 6.2 are computationally indistinguishable for any* i^*.

Proposition 6.4. *By the* LWE *assumption, the output from* $\mathsf{NewSetup}_{i^*}$ *in Algorithm 3.1 and the output from* $\mathsf{NewNewSetup}_{i^*}$ *in Algorithm 6.2 are computationally indistinguishable for any* i^*.

Now, we can prove single-bit hiding property

Theorem 6.5. *Algorithm 3.1 satisfies single-bit hiding property*

Proof. Suppose in a single-bit hiding game, the adversary $\mathcal{A}$ query i^*. By Proposition 6.3, 6.4, the system is computationally indistinguishable to the system with $\mathsf{NewNewSetup}_{i^*}$. So we just need to show that under this system, r_{i^*} is statistically close to random.

We first consider $\mathbf{t}$. The information about $\mathbf{t}$ can only be leaked through $\{\pi_i\}_{i\neq i^*}, \mathsf{com}$, where we know that $\pi_i = \mathbf{W}_i\mathbf{t}, \mathsf{com} = \mathbf{U}\mathbf{t}$. Since $\mathbf{W}_i \in \tilde{D}_\sigma^{m\times L}, \mathbf{U} \in \mathbb{Z}_q^{n\times L}$, both $\mathbf{W}_i$'s and $\mathbf{U}$ have $k \cdot poly(\lambda)$ times more columns than the rows, therefore $\mathbf{t}$ has a high min-entropy conditioned on $\mathsf{com}, \{\pi_i\}_{i\neq i^*}$ given that $\mathbf{t}$ has high length of L and sampled uniformly. Therefore, this completely hides $\mathbf{t}$.

Note that $r_{i^*} = \mathsf{round}(\mathbf{v}_{i^*}^\top \pi_{i^*}^\top + d_{i^*})$, where d_{i^*} is known, hence we will show $\mathbf{v}_{i^*}^\top \pi_{i^*}^\top$ is statistically close to random. We have

$$\mathbf{v}_{i^*}^\top \pi_{i^*}^\top = \mathbf{v}_{i^*}^\top \mathbf{W}_{i^*}\mathbf{t}$$

Since $\mathbf{v}_{i^*}^\top \in \mathbb{Z}_q^m$ is random, and for each j^{th} column $\mathbf{w}_j$ of $\mathbf{W}_{i^*}$ satisfies $\mathbf{w}_j \in \tilde{D}_\sigma^m$, we have $\mathbf{v}_{i^*}^\top \mathbf{W}_{i^*}$ is statistically close to uniform as each entry is statistically close to uniform. Now, since $\mathbf{t}$ has a high min-entropy, and is sampled uniformly in $[-B, B]^L$ hence $\mathbf{v}_{i^*}^\top \pi_{i^*}^\top$ is statistically close to random. Therefore, adding a constant and rounds to obtain r_{i^*} still preserves randomness. □

Remark 6.6. $\mathbf{t}$ *is assumed to be in* $\{0,1\}^L$ *in [BCD+25], where we can certainly adapt this change in our hidden bit generator construction, this will change the value of* TestBound *and* RoundBound. *In the end* $\mathbf{v}_{i^*}^\top \mathbf{W}_{i^*}\mathbf{t}$ *is random because of the leftover hash lemma.*

References

[BCD+25] Pedro Branco, Arka Rai Choudhuri, Nico Döttling, Abhishek Jain, Giulio Malavolta, and Akshayaram Srinivasan. Black-Box Non-Interactive Zero Knowledge from Vector Trapdoor Hash. In EUROCRYPT, pages 64–92. Springer, 2025.

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

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

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In CRYPTO, pages 186–194, 1986.

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

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In STOC, pages 197–206, 2008.

[KMY20] Fuyuki Kitagawa, Takahiro Matsuda, and Takashi Yamakawa. NIZK from SNARG. In TCC, 2020

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

[Wat24] Brent Waters. A new approach for non-interactive zero-knowledge from learning with errors. In STOC, pages 399–410. ACM, 2024.

Appendix

Proof of Proposition 6.3. Suppose there is a PPT algorithm $\mathcal{A}$ that is able to distinguish Setup and $\mathsf{NewSetup}_{i^*}$. We will create an algorithm $\mathcal{B}$ that distinguish the two distribution from the trapdoor pre-sampling.

Suppose the challenger gives $\mathbf{A}^*, \mathbf{W}^*, \mathbf{U}^*, \mathcal{B}$ would run Setup algorithm under the exception that the following variables are fixed $\mathbf{A}_{i^*} = \mathbf{A}^*, \mathbf{W}_{i^*} = \mathbf{W}^*, \mathbf{U} = \mathbf{U}^*$. Then, $\mathcal{B}$ sends the output to $\mathcal{A}$.

If the challenger obtains its value from the distribution,

$$(\mathbf{A}, \mathbf{W} \leftarrow \mathsf{TrapPreSamp}(\mathbf{A}, \mathsf{td}, \mathbf{U}), \mathbf{U} \leftarrow \mathbb{Z}_q^n)$$

then the input given to $\mathcal{A}$ is identical in distribution to the Setup algorithm from Algorithm 3.1.

If the challenger obtains its value from the other distribution, namely

$$(\mathbf{A} \in \mathbb{Z}_q^{n \times m}, \mathbf{W} \leftarrow \tilde{D}_\sigma^{m \times L}, \mathbf{U} = \mathbf{AW})$$

then the input given to $\mathcal{A}$ is identical in distribution to the $\mathsf{NewSetup}_{i^*}$ algorithm from Algorithm 6.1.

Hence, if $\mathcal{A}$ is able to distinguish which setup algorithm is used. Then, $\mathcal{B}$ can win the trapdoor pre-sampling game by returning the corresponding distribution from trapdoor pre-sampling based on the output from $\mathcal{A}$. □

Proof of Proposition 6.4. Suppose there is a PPT algorithm $\mathcal{A}$ that is able to distinguish the algorithms $\mathsf{NewSetup}_{i^*}, \mathsf{NewNewSetup}_{i^*}$. We will create an algorithm $\mathcal{B}$ that solves the Decisional LWE.

Suppose the challenger gives $\mathbf{A}^*, \mathbf{v}^*, \mathcal{B}$ would run Setup algorithm under the exception that the following variables are fixed $\mathbf{A}_{i^*} = \mathbf{A}^*, \mathbf{v}_{i^*} = \mathbf{v}^*$. Then, $\mathcal{B}$ sends the output to $\mathcal{A}$.

If the challenger obtains its value from the distribution, where $\mathbf{s} \in \mathbb{Z}_q^n, \mathbf{e} \in \tilde{D}_\sigma^m$

$$(\mathbf{A} \in \mathbb{Z}_q^{n \times m}, \mathbf{s}^\top A + \mathbf{e}^\top)$$

then the input given to $\mathcal{A}$ is identical in distribution to the $\mathsf{NewSetup}_{i^*}$ algorithm from Algorithm 6.1.

If the challenger obtains its value from the other distribution, namely

$$(\mathbf{A} \in \mathbb{Z}_q^{n \times m}, \mathbf{u} \in \mathbb{Z}_q^m)$$

then the input given to $\mathcal{A}$ is identical in distribution to the $\mathsf{NewNewSetup}_{i^*}$ algorithm from Algorithm 6.2.

Hence, if $\mathcal{A}$ is able to distinguish which setup algorithm is used. Then, $\mathcal{B}$ can win the decisional LWE by returning the corresponding distribution from decisional LWE based on the output from $\mathcal{A}$. □