A New Approach for Non-Interactive Zero-Knowledge from Learning with Errors

Brent Waters
UT Austin and NTT Research
bwaters@cs.utexas.edu

Abstract

We put forward a new approach for achieving non-interactive zero-knowledge proofs (NIKZs) from the learning with errors (LWE) assumption (with subexponential modulus to noise ratio). We provide a LWE-based construction of a hidden bits generator that gives rise to a NIZK via the celebrated hidden bits paradigm. A notable feature of our construction is its simplicity. Our construction employs lattice trapdoors, but beyond that uses only simple operations. Unlike prior solutions, we do not rely on a correlation intractability argument nor do we utilize fully homomorphic encryption techniques. Our solution provides a new methodology that adds to the diversity of techniques for solving this fundamental problem.

1 Introduction

Zero-knowledge proofs [GMR85] enable a prover to convince a verifier of the validity of a statement without revealing anything about why the statement holds. Of particular interest and power are non-interactive zero-knowledge proofs [BFM88] (NIZKs). NIZKs allow a prover with the aid of a common reference string (CRS) to non-interactively produce a convincing proof that can be shared with any number of verifiers. While it has long been known [GMW86] how to achieve zero-knowledge proofs for $\mathcal{NP}$ from one way functions using interactive provers, achieving NIZKs from standard cryptographic assumptions has been a notoriously challenging endeavor.

The first realization of non-interactive zero-knowledge for $\mathcal{NP}$ relations is due to Feige, Lapidot and Shamir [FLS90]. The authors show how to create a non-interactive zero-knowledge proof in an idealized model called the hidden bits model. In this model the prover has access to a set of randomly chosen bits that are hidden to the rest of the world. As part of its proof, the prover can selectively choose which bits to reveal, but is not allowed to modify them. FLS described how to create NIZKs for $\mathcal{NP}$ in the hidden bits model and provided a concrete construction realizing the model assuming the difficulty of factoring. (Their construction was built from certified trapdoor permutations which are known to exist based on the difficulty of factoring.)

For more than a decade thereafter, the only number-theoretic realizations of general NIZKs were factoring based. However starting with the seminal work of Boneh and Franklin [BF01], the community witnessed an explosion in a number of new cryptographic applications driven by utilizing groups with efficiently computable bilinear maps. In 2003 Canetti, Halevi and Katz [CHK03] described a NIZK based on the search Diffie-Hellman problem in bilinear groups. Their construction leveraged the aforementioned hidden bits paradigm. Subsequently, Groth, Ostrovsky and Sahai [GOS06] introduced a direct gate by gate NIZK solution for circuit satisfiability in bilinear groups. Thus by the early 2000s, researchers had a second number-theoretic tool for building NIZKs for $\mathcal{NP}$ firmly in hand.

By the early 2010s, the Learning with Errors (LWE) problem [Reg05] was emerging as the next number-theoretic powerhouse for developing cryptographic functionality. Innovations from LWE around the time include Identity-Based Encryption [GPV08, CHKP10, ABB10a], fully homomorphic encryption [Gen09, BV11] and attribute-based encryption for circuits [GVW13]. Given that many of these new LWE-based cryptographic functionalities met or exceeded what was achieved from bilinear maps a decade earlier, one might have predicted that non-interactive zero-knowledge proofs would soon be added to the LWE achievement list. In particular researchers might have expected a

learning with errors analog of either a NIZK based on the hidden bits paradigm or a more direct construction in the Groth et al. [GOS06] style.

Despite such expectations, finding a hidden bits model construction from LWE remained elusive. Researchers began to focus on a very different approach to building proof systems through the Fiat-Shamir heuristic [FS86]. The Fiat-Shamir heuristic is used to transform public coin interactive verification protocols by replacing verifier messages with hashes of the transcript so far. While the Fiat-Shamir heuristic is traditionally analyzed in the random oracle model, one can instead try to achieve a standard model analysis by identifying a concrete "correlation intractable" [CGH04] property of the hash function. A hash function $H(\cdot)$ is *correlation intractable* for a relation $\mathcal{R}$ if it is computationally hard to find an x such that $\mathcal{R}(x, H(x))$ holds. In the context of three round proof systems, one can think of x as the prover's first message and $H(x)$ as the verifier's response, where the relation $\mathcal{R}$ captures the verifier responses a prover could cheat on (e.g., it should be hard to find an x such that cheating is possible).

Initial works [KRR17, HL18, CCRR18] in this vein made substantial progress, but depended on non-standard assumptions. A breakthrough occurred when Canetti, Chen, Holmgren, Lombardi, Rothblum, Rothblum and Wichs [CCH+19] showed how to realize correlation intractabilty for search relations from circular-secure fully homomorphic encryption (FHE). Next, Peikert and Shiehian [PS19] showed how to realize the framework from the learning with errors assumption by replacing circular security with a clever "inert commitment" step. This finally resolved the question of building NIZKs from LWE. Remarkably, the concept of correlation intractability continued to bear fruit. Brakerski, Koppula, and Mour [BKM20] introduced a form of approximate correlation intractability and showed how to build NIZKs from the combination of the learning parity with Noise (LPN) and decisional Diffie-Hellman (DDH) assumptions. Subsequently, Jain and Jin [JJ21] gave a solution from subexponentially-hard DDH (over groups with certain complexity conditions on the group operation) also using correlation intractability.

While these achievements from correlation intractability are considerable, many researchers felt there remained gaps in our understanding and lingering questions. In particular it was unresloved why no one had been able to build NIZKs from the longer established (in the standard model) hidden bits methodology. Was there a fundamental reason why or did the community simply miss a solution? Was the utilization of correlation intractability or fully homomorphic encryption techniques somehow inherent in building NIZKs from LWE? Do all NIZK solutions from LWE need to make some type of non-black box access to public key decryption along the lines of [CCH+19, PS19]?

Our Contribution

We propose a hidden bits construction of NIZKs from the LWE assumption (with subexponential modulus to noise ratio) via the hidden bit generator abstraction [QRW19]. The construction does not utilize correlation intractable hashes and does not leverage the fully homomorphic techniques of [GSW13] or similar works. It also does not require non-black box access to other cryptographic primitives. A notable feature of our construction is its simplicity. It employs lattice trapdoors [GPV08], but beyond that uses only simple operations. Thus, this work helps resolve some of the lingering questions on NIZKs from LWE and contributes to the diversity of approaches to building NIZKs.

On the Importance of Diversity in Approaches to NIZKs

Before detailing our technical approach, we put forward arguments as to why it is critical to have a diversity of solutions to a problem as fundamental as achieving NIZKs from LWE.

First, realizing multiple approaches to solving a problem is central to obtaining a comprehensive understanding. Consider for comparison the problem of obtaining chosen ciphertext (CCA) security [RS91] from LWE. This was an open problem first solved by Peikert and Waters [PW08] with the introduction of lossy trapdoor functions. However, since then many ways of attacking this problem have emerged including: (1) direct applications of lattice trapdoors [Pei09], (2) the BCHK [BCHK07] identity-based encryption (IBE) to CCA transformation applied to an appropriate IBE system [CHKP10, ABB10a], (3) applying a chosen plaintext attack (CPA) to CCA [NY90] transformation from LWE-based NIZKs [PS19] and (4) using an LWE-based hinting pseudorandom generator [KW19]. The ideas behind each of these are quite diverse and put together they create a clearer picture of the connection between CCA security and the LWE problem.

Second, we expect different approaches to NIZKs to result in advances in other areas and problems. For instance, there is a tight connection between succinct batch arguments (BARGs) and NIZKs. In one direction batch arguments (with certain caveats) imply NIZKs [CW23, BKP+24, BWW23]. In the other direction, many of the techniques behind

building batch proof systems [CJJ21a, CJJ21b, WW22, HJKS22, CGJ+23] are rooted in those developed for NIZKs including correlation resistance. Most of the above BARG results rely on probabilistically-checkable proof type techniques for building batch arguments; one exception to this is is the work of Waters and Wu [WW22] who present a bilinear map solution with a more direct route to circuit satisfiability. Arguably, this difference is a reflection of the connection between Waters and Wu [WW22] and the Groth et al. [GOS06] NIZK system which itself is direct and does not rely on correlation-resistant hashes. We believe pushing for a diversity in approaches to NIZKs will result in a diversity in batch proof systems and a host of additional applications.

Finally, exploring a variety of methods to solve NIZKs from LWE (and other assumptions) can potentially lead to more practically efficient systems. Our approach makes some progress in this direction by removing the need to homomorphically evaluate a public key decryption operation. At the same time, we observe that the use of a superpolynomial-sized modulus and the overhead from the hidden bits transformation is limiting from a practical perspective. However, we believe opening the door to new constructions is important especially with an eye on potential future direct constructions such as an analog in the style of Groth et al. [GOS06].

1.1 Our Approach

We present a construction of a hidden bits generator abstraction proposed by Quach, Rothblum and Wichs [QRW19]. This is known to imply a non-interactive zero knowledge proof [FLS90]. Our exposition will focus on the construction and proof of the hidden bits generator.

A hidden bits generator consists of three algorithms: $(\mathsf{Setup}, \mathsf{GenBits}, \mathsf{Verify})$. The first algorithm is a randomized setup algorithm $\mathsf{Setup}(1^\lambda, 1^k) \to \mathsf{crs}$ which takes as input the security parameter λ and a length parameter k and outputs a common reference string crs. The second is a randomized algorithm $\mathsf{GenBits}(\mathsf{crs}) \to (\mathsf{com}, \mathbf{r}, (\pi_1, \ldots, \pi_k))$ that takes as input the crs and outputs a short commitment com to a k length bitstring $\mathbf{r} \in \{0,1\}^k$ along with k proofs $(\pi_1, \ldots, \pi_k)$ which can be used to prove the respective output bits relative to com. Finally, there is a deterministic verification algorithm $\mathsf{Verify}(\mathsf{crs}, \mathsf{com}, i, \beta, \pi) \to b$ that verifies the proof π of the opening of a bit β for index i relative to the commitment com.

A hidden bits generator has two main properties. The first is a form of binding security. Each crs is associated with a set $\mathcal{V}^{\mathsf{crs}}$. It must be the case that, except with negligible probability over the choice of crs, no attacker can successfully open a subset of bits to a string that is not a substring of some $\mathbf{r} \in \mathcal{V}^{\mathsf{crs}}$. In addition, we want $\mathcal{V}^{\mathsf{crs}}$ to be somewhat small and certainly much smaller than 2^k — the number of possible k-length strings. More precisely, there should be a fixed polynomial p and constant $\nu < 1$ where $|\mathcal{V}^{\mathsf{crs}}| \leq 2^{k^\nu \cdot p(\lambda)}$. Second, the hidden bits generator should exhibit a hiding property. Consider a computationally bounded attacker that is given the bits and corresponding proofs $\{\mathbf{r}_i, \pi_i\}_{i \in I}$ for some set $I \subseteq [k]$ of its choice. The remaining bits should remain hidden and indistinguishable from random.

Our Hidden Bits Generator Construction

The best way to explain our construction is to jump into describing it. For security parameter λ, we use a prime modulus q where $2^\lambda < q < 2^{\lambda+1}$. We use the typical LWE parameters n, m, σ which will be set to be some appropriate polynomial in λ. In addition, we set $L = \lambda mk + 5\lambda + 1$ where k is given as input to the setup. We use $\tilde{D}_{\mathbb{Z},\sigma}$ to denote the distribution of a truncated discrete Gaussian of width σ (see Section 3). We also employ lattice trapdoors [Ajt96, GPV08, ABB10b, ABB10a, CHKP10, MP12] with algorithms $(\mathsf{TrapGen}, \mathsf{SamplePre})$ (see Theorem 3.4). We first describe the setup algorithm.

$\mathsf{Setup}(1^\lambda, 1^k) \to \mathsf{crs}$:

1. Run $(\mathbf{A}_i, \mathsf{td}_i) \leftarrow \mathsf{TrapGen}(1^n, q, m)$ for all $i \in [k]$.
2. Choose random $\mathbf{U} \xleftarrow{R} \mathbb{Z}_q^{n \times L}$.
3. Sample
   (a) $\mathbf{s}_i \xleftarrow{R} \mathbb{Z}_q^n$ for all $i \in [k]$
   (b) $\mathbf{e}_i \xleftarrow{R} \tilde{D}_{\mathbb{Z},\sigma}^m$ for all $i \in [k]$

(c) $d_i \xleftarrow{R} \mathbb{Z}_q$ for all $i \in [k]$

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

The setup algorithm creates k different matrices $\mathbf{A}_1, \ldots, \mathbf{A}_k$ with corresponding trapdoors using TrapGen. Then it samples a random matrix $\mathbf{U}$ of dimension $n \times L$. and uses the SamplePre algorithm to create k matrices $\mathbf{W}_1, \ldots, \mathbf{W}_k$ such that $\mathbf{A}_i \mathbf{W}_i = \mathbf{U}$ for all $i \in [k]$. Looking forward this structure will allow us to create a proof for each bit that resolves to the same commitment. We observe that the public parameters grow as $k^2 \cdot \mathsf{poly}(\lambda)$ and will later see how a quadratic growth in k is intertwined with our hiding analysis. The next algorithm is GenBits which we give below.

$\mathsf{GenBits}(\mathsf{crs}) \rightarrow (\mathsf{com}, \mathbf{r}, (\pi_1, \ldots, \pi_k))$:

1. Sample $\mathbf{t} \xleftarrow{R} [-2^{.5\lambda}, 2^{.5\lambda}]^L$.
2. Compute $\pi_i = \mathbf{W}_i \mathbf{t}$ for all $i \in [k]$.
3. Set $r_i = \left\lfloor \mathbf{v}_i^\intercal \pi_i + d_i \right\rceil$ for all $i \in [k]$.
4. Set $\mathsf{com} = \mathbf{U}\mathbf{t}$.
5. Output $(\mathsf{com}, \mathbf{r}, (\pi_1, \ldots, \pi_k))$.

The GenBits algorithm is rather simple and straightforward. The process is to first choose a vector $\mathbf{t}$ with entries randomly chosen in the interval $[-2^{.5\lambda}, 2^{.5\lambda}]$. Both the commitment and proofs are generated by simply multiplying corresponding crs parameters by $\mathbf{t}$. Finally, to get the i-th output bit, first multiply $\mathbf{v}_i^\intercal \pi_i$, then add a scalar d_i and finally round the result. We observe that no fully homomorphic encryption type techniques are utilized. The final algorithm is verification.

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

1. Check if $\|\pi\|_\infty \leq \mathtt{TestBound}$; output 0 if this does not hold, where $\mathtt{TestBound} = \sigma\sqrt{\lambda} \cdot L \cdot 2^{.5\lambda}$.
2. Check if $\mathsf{com} \stackrel{?}{=} \mathbf{A}_i \pi$; output 0 if this does not hold.
3. Check if $\beta \stackrel{?}{=} \left\lfloor \mathbf{v}_i^\intercal \pi + d_i \right\rceil$; output 0 if does not hold.
4. Finally, check if $\beta \stackrel{?}{=} \left\lfloor \mathbf{v}_i^\intercal \pi + d_i + \mathtt{RoundingBound} \right\rceil$ and $\beta \stackrel{?}{=} \left\lfloor \mathbf{v}_i^\intercal \pi + d_i - \mathtt{RoundingBound} \right\rceil$; output 1 if it holds and 0 otherwise. Here we set $\mathtt{RoundingBound} = \sigma\sqrt{\lambda} \cdot m \cdot \mathtt{TestBound}$.

Again, all operations are simple matrix multiplications along with rounding and bounds checks. To see that correctness holds we first observe that any correctly generated π from GenBits will satisfy the bounds check of Step 1 of verification by the setting of parameters. Second, consider $\pi_i = \mathbf{W}_i \mathbf{t}$ generated by the GenBits algorithm. The verification checks $\mathsf{com} \stackrel{?}{=} \mathbf{A}_i \pi_i$. However, we have that $\mathbf{A}_i \pi_i = \mathbf{A}_i \mathbf{W}_i \mathbf{t}$ by the computation of π. And that $\mathbf{A}_i \mathbf{W}_i \mathbf{t} = \mathbf{U}\mathbf{t}$ by the programming of $\mathbf{W}$. In addition, $\mathbf{U}\mathbf{t} = \mathsf{com}$ from Step 4 of the GenBits algorithm, so the commitment test will pass. Finally, we see that the final check of verification rejects if the value of $\mathbf{v}_i^\intercal \pi + d_i$ is within $\mathtt{RoundingBound}$ of one of the the two rounding thresholds. Since $\mathtt{RoundingBound}$ is $2^{.5\lambda}\mathsf{poly}(\lambda)$ this causes a correctness error with at most negligible probability. In Section 4.1, we show a generic method to remove the correctness error entirely.

Binding Security
To see why the binding property holds for our hidden bits generator, it helps to first pretend that the noise vectors $\mathbf{e}_i$ from setup were all set to $\mathbf{0}$. Imagine a proof π_i that successfully verifies for a particular index i and commitment value com. Then the output bit is $\lfloor \mathbf{v}_i^\intercal \pi_i + d_i \rceil$. We observe that

$$\mathbf{v}_i^\intercal \pi_i + d_i = (\mathbf{s}_i^\intercal \mathbf{A}_i + \mathbf{e}_i^\intercal)\pi_i + d_i = \mathbf{s}_i^\intercal \mathbf{A}_i \pi_i + \mathbf{e}_i^\intercal \pi_i + d_i = \mathbf{s}_i^\intercal \mathsf{com} + \mathbf{e}_i^\intercal \pi_i + d_i$$

These equations follow from the fact that $\mathbf{A}_i \pi_i = \mathsf{com}$ if the proof verifies. If we make the temporary presumption that $\mathbf{e}_i = \mathbf{0}$, then we have $\mathbf{s}_i^\intercal \mathsf{com} + \mathbf{e}_i^\intercal \pi_i + d_i = \mathbf{s}_i^\intercal \mathsf{com} + d_i$. In this case the commitment vector com completely determines the output bit! Since the commitment is of roughly $\lambda \cdot n$ bits, we also get our succinctness condition. While it is tempting to just remove the $\mathbf{e}_i$ vectors, this noise in setup will be necessary to argue hiding. Thus the term $\mathbf{e}_i^\intercal \pi_i$ threatens to hurt binding as different π_i values could potentially flip which output bit $\mathbf{r}_i$ is accepted.

On closer inspection the largest (absolute) value that $\mathbf{e}_i^\intercal \pi_i$ could take on is not much more than $2^{\cdot 5\lambda}$ and is in fact less then the `RoundingBound` parameter used in the final verification step. This is due to the fact that $\mathbf{e}_i$ is of low norm and the verification size restrictions on accepting π_i. Therefore this flipping attack is only possible if $\mathbf{s}_i^\intercal \mathsf{com} + d_i$ is within (around) `RoundingBound` of one of the two rounding threshold values. However, if this is the case then the final check of the verification will detect this and reject.

Hiding Security We now move to proving hiding security of our construction. We do so by considering a simplified hiding game which we call single bit hiding. In this game the attacker specifies a single challenge index $i^* \in [k]$. The challenger runs GenBits and gives the attacker com and $(\mathbf{r}_i, \pi_i)$ for all $i \neq i^*$. The attacker must then guess $\mathbf{r}_{i^*}$. This notion implies the normal hiding game via a straightforward hybrid argument. To prove security we first introduce an alternative setup algorithm called SetupHiding in which an output bit r_{i^*} will be statistically hidden even given all the information above.

$\mathsf{SetupHiding}(1^\lambda, 1^k) \rightarrow$ crs:

1. Run $(\mathbf{A}'_i, \mathsf{td}_i) \leftarrow \mathsf{TrapGen}(1^{n+1}, q, m)$ and parse $\mathbf{A}'_i = \begin{bmatrix} \mathbf{A}_i \\ \mathbf{v}_i^\intercal \end{bmatrix}$ where $\mathbf{A}_i \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{v}_i$ is a vector for all $i \in [k]$.
2. Choose random $\mathbf{U} \xleftarrow{R} \mathbb{Z}_q^{n \times L}$.
3. Sample
   (a) $\mathbf{u}_i \xleftarrow{R} \mathbb{Z}_q^L$ for all $i \in [k]$
   (b) $d_i \xleftarrow{R} \mathbb{Z}_q$ for all $i \in [k]$
4. Set $\mathbf{U}'_i = \begin{bmatrix} \mathbf{U} \\ \mathbf{u}_i^\intercal \end{bmatrix}$ for all $i \in [k]$.
5. Sample $\mathbf{W}_i \xleftarrow{R} \mathsf{SamplePre}(\mathbf{A}'_i, \mathsf{td}_i, \mathbf{U}'_i, \sigma)$ for all $i \in [k]$.
6. Output $\mathsf{crs} = \{\mathbf{U}, \mathbf{A}_i, \mathbf{W}_i, \mathbf{v}_i, d_i\}_{i \in [k]}$.

We can prove the output of this alternative setup is indistinguishable from the original by a sequence of hybrids using the LWE assumption along with statistical properties of lattice trapdoors. This is detailed in Section 6.

We next prove hiding security in this mode by establishing (with high probability) the existence of a *short* vector $\mathbf{c}$ where $\mathbf{W}_i \mathbf{c} = \mathbf{0}$ for all $i \neq i^*$, but $\mathbf{u}_{i^*}^\intercal \mathbf{c} = \lfloor q/2 \rfloor$. To this end we first give a counting argument where we note that the number of binary vectors of length L is 2^L which is significantly larger than the number of outputs that come from multiplying a binary vector of length L by *all* of the different matrices $\{\mathbf{W}_i\}_{i \neq i^*}$. We can bound the latter by $2^{(\lambda+1)mk}$. This means that there must exist two distinct binary vectors $\mathbf{y}_1, \mathbf{y}_0 \in \{0,1\}^L$ such that $\mathbf{W}_i \mathbf{y}_0 = \mathbf{W}_i \mathbf{y}_1$ for all $i \neq i^*$. This implies a nonzero vector $\mathbf{h} = \mathbf{y}_0 - \mathbf{y}_1 \in \{-1, 0, 1\}$ where $\mathbf{W}_i \mathbf{h} = \mathbf{0}$ for all $i \neq i^*$. We can actually give a variation of this argument that establishes the existence of several vectors $\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_{5\lambda} \in \{-1, 0, 1\}^L$ that are linearly independent and where $\mathbf{W}_i \mathbf{h}_j = \mathbf{0}$ for $i \neq i^*$ and $j \in [5\lambda]$. We defer the extension of this analysis to the

main body. Since the vectors $\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_{5\lambda}$ are linearly independent and $\mathbf{u}_{i^*}$ is chosen uniformly at random (and independently of $\{\mathbf{W}_i\}_{i \neq i^*}$) we can also show (again deferred to main body) via the leftover hash lemma that with high probability there exists a linear combination of these with binary coefficients that gives us a vector $\mathbf{c}$ where $\mathbf{c}$ is short and $\mathbf{u}_{i^*}^\intercal \mathbf{c} = \lfloor q/2 \rfloor$. We emphasize that we only need to prove the existence of such a vector; an inefficient algorithm to find it is sufficient as our proof will be statistical from here on out.

With this vector in hand, we can return to our proof of hiding security. We next consider a security game where the challenger in the GenBits algorithm chooses a random $\delta \in \{0,1\}$ and $\mathbf{t} \in [-2^{.5\lambda}, 2^{.5\lambda}]$ and uses $\mathbf{t} + \delta\mathbf{c}$ as the random vector for the algorithm instead of just $\mathbf{t}$ as before. This distribution is statistically close to the previous one by the Smudging Lemma (see Lemma 3.2) due to the shortness of $\mathbf{c}$.

Now an interesting property emerges. Whether δ is 0 or 1 has no bearing on the output on the commitment com, proofs π_i or output bits $\mathbf{r}_i$ for all $i \neq i^*$. This is due to the fact that $\mathbf{W}_i \mathbf{c} = \mathbf{0}$ for all $i \neq i^*$. But the output value $\mathbf{r}_{i^*}$ will either be flipped or not depending on whether δ is 0 or 1.

$$
\begin{aligned}
\left\lfloor \mathbf{v}_{i^*}^\intercal \pi_{i^*} + d_{i^*} \right\rceil &= \left\lfloor \mathbf{v}_{i^*}^\intercal \mathbf{W}_{i^*} (\mathbf{t} + \delta\mathbf{c}) + d_{i^*} \right\rceil && \text{By definition of the game.} \\
&= \left\lfloor \mathbf{u}_{i^*}^\intercal (\mathbf{t} + \delta\mathbf{c}) + d_{i^*} \right\rceil && \text{Since } \mathbf{u}_{i^*}^\intercal = \mathbf{v}_{i^*}^\intercal \mathbf{W}_{i^*} \text{ in the game.} \\
&= \left\lfloor \mathbf{u}_{i^*}^\intercal \mathbf{t} + d_{i^*} + \delta \lfloor q/2 \rfloor \right\rceil && \text{Since } \mathbf{u}_{i^*}^\intercal \mathbf{c} = \lfloor q/2 \rfloor. \\
&= \left\lfloor \mathbf{u}_{i^*}^\intercal \mathbf{t} + d_{i^*} \right\rceil \oplus \delta && \text{With all but } 1/q \text{ probability.}
\end{aligned}
$$

Since δ does not appear anywhere else in the proof, the bit is hidden.

Stepping back we remark that we introduce an interesting *vector emergence* technique. By making the parameter L long enough, we argue the existence of a vector $\mathbf{c}$ with certain properties even though we never explicitly targeted such properties in our choice of the random matrix $\mathbf{U}'_i$. The lack of explicit targeting was helpful in switching between modes. We combine this with the fact that $\mathbf{c}$ has small coefficients compared to $\mathbf{t}$ to make the bit flipping vector emerge in our proof.

We conclude by remarking that our approach has a dual mode property where the default setup could be either hiding or binding. This results in a system where we could make the compiled NIZK have statistical soundness or statistical zero-knowledge.

2 Hidden Bits Generators

We will employ the hidden bits generator abstraction first introduced by Quach, Rothblum and Wichs [QRW19] where different variations appeared in other works [KMY20, CW23]. A hidden bits generator Π_{HBG} consists of three algorithms Setup, GenBits and Verify. The Setup algorithm takes as input the security parameter λ and a length parameter k and outputs a common reference string crs. Once the crs is established a prover can run the randomized GenBits(crs) algorithm and obtain a commitment com, string of k bits $\mathbf{r}$ and individual proofs $(\pi_1, \ldots, \pi_k)$ for each bit. The Verify algorithm can verify the individual bits as $\mathsf{Verify}(\mathsf{crs}, \mathsf{com}, i, \beta, \pi) = 1$.

The binding property of a Π_{HBG} system requires that each crs will be associated with a set $\mathcal{V}^{\mathsf{crs}} \subseteq \{0,1\}^k$. We require that if a (computationally unbounded) adversary successfully opens a substring of bits for some commitment com, then the substring must be consistent with at least one $\mathbf{r} \in \mathcal{V}^{\mathsf{crs}}$. Moreover, $\mathcal{V}^{\mathsf{crs}}$ will be relatively small. For some constant $\nu < 1$ and fixed polynomial p we have $|\mathcal{V}^{\mathsf{crs}}| \leq 2^{m^\nu \cdot p(\lambda)}$.

To prove hiding security we will define a notion of single point hiding. In this definition an attacker will first selectively specify an index i^* and the challenger will generate the crs, commitment com and give out bit values and proofs $\mathbf{r}_i, \pi_i$ for all $i \neq i^*$. It should be computationally hard for an attacker to distinguish r_{i^*} from a random bit. We then show that this single point hiding property actually implies via a simple hybrid argument the adaptive notion of hiding. In the standard hiding game the attacker requests revelations for a set I in one go and needs to distinguish $\mathbf{r}_{\bar{I}}$ from a random string. While this strengthens prior works which considered a selective notion, the main utility is that it allows us to simplify our hiding proof by focusing on one bit at a time.

Notation We write λ to denote the security parameter. For a positive integer $n \in \mathbb{N}$, we write $[n]$ to denote the set $\{1, \ldots, n\}$. We write $\mathsf{poly}(\lambda)$ to denote a fixed function that is $O(\lambda^c)$ for some $c \in \mathbb{N}$ and $\mathsf{negl}(\lambda)$ to denote a function

that is $o(\lambda^{-c})$ for all $c \in \mathbb{N}$. We say an event occurs with overwhelming probability if its complement occurs with negligible probability. We say an algorithm is efficient if it runs in probabilistic polynomial time in the length of its inputs. Notationally, for a bitstring $\mathbf{r} \in \{0,1\}^n$ and a set of indices $I \subseteq [n]$, we write $\mathbf{r}_I \in \{0,1\}^{|I|}$ to denote the substring corresponding to the bits of $\mathbf{r}$ indexed by I.

Definition 2.1 (Hidden-Bits Generator)**.** A hidden-bits generator a tuple of efficient algorithms $\Pi_{\mathsf{HBG}} = (\mathsf{Setup}, \mathsf{GenBits}, \mathsf{Verify})$ with the following syntax:

- $\mathsf{Setup}(1^\lambda, 1^k) \to \mathsf{crs}$: On input the security parameter λ, and the output length k, the setup algorithm outputs a common reference string crs.
- $\mathsf{GenBits}(\mathsf{crs}) \to (\mathsf{com}, \mathbf{r}, (\pi_1, \ldots, \pi_k))$: On input of the common reference string crs, the generator algorithm outputs a string $\mathbf{r} \in \{0,1\}^k$ and a tuple of proofs $\pi_1, \ldots, \pi_k$.
- $\mathsf{Verify}(\mathsf{crs}, \mathsf{com}, i, \beta, \pi) \to b$: On input a common reference string crs, an index i, a bit $\beta \in \{0,1\}$, and a proof π, the verification algorithm outputs a bit $b \in \{0,1\}$.

We require Π_{HBG} to satisfy the following properties:

- **Correctness:** For all $\lambda, k \in \mathbb{N}$ and all indices $i \in [k]$, we have

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

- **Statistical binding:** For every crs in the support of the algorithm $\mathsf{Setup}(1^\lambda, 1^k)$, there exists a set $\mathcal{V}^{\mathsf{crs}}$ with the following properties:

  (i) **Output sparsity.** There exists a universal constant $\nu < 1$ and a fixed polynomial $p(\cdot)$ such that for every polynomial $k = k(\lambda)$ there exists a λ_0 such that for all $\lambda > \lambda_0$ and for every crs in the support of $\mathsf{Setup}(1^\lambda, 1^k)$, $|\mathcal{V}^{\mathsf{crs}}| \leq 2^{k^\nu \cdot p(\lambda)}$.

  (ii) **Statistical binding.** For a security parameter λ, we define the statistical binding game between an computationally unbounded adversary $\mathcal{A}$ that and a challenger as follows:

  (a) On input the security parameter 1^λ, algorithm $\mathcal{A}$ starts by outputting the length parameter 1^k.
  (b) The challenger samples $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^k)$ and gives crs to $\mathcal{A}$.
  (c) Algorithm $\mathcal{A}$ outputs a tuple $(\mathsf{com}, I \subseteq [k], \mathbf{r}_I, \{\pi_i\}_{i \in I})$.
  (d) The output of the experiment is $b = 1$ if $\mathbf{r}_I \notin \mathcal{V}_I^{\mathsf{crs}}$ and $\mathsf{Verify}(\mathsf{crs}, \mathsf{com}, i, \mathbf{r}_i, \pi_i) = 1$ for all $i \in I$, where $\mathcal{V}_I^{\mathsf{crs}} := \{\mathbf{r}_I : \mathbf{r} \in \mathcal{V}^{\mathsf{crs}}\}$. Otherwise, the output is $b = 0$.

  We say the Π_{HBG} is statistically binding if for all (not necessarily) efficient adversaries $\mathcal{A}$, $\Pr[b = 1] = \mathsf{negl}(\lambda)$ in the statistical binding security game.

- **Adaptive Computational hiding:** For a security parameter λ and bit $b \in \{0,1\}$, we define the adaptive computational hiding game between an adversary $\mathcal{A}$ and a challenger as follows:

  1. On input the security parameter 1^λ, algorithm $\mathcal{A}$ starts by outputting the length parameter 1^k.
  2. The challenger samples $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^k)$, $(\mathsf{com}, \mathbf{r}, (\pi_1, \ldots, \pi_k)) \leftarrow \mathsf{GenBits}(\mathsf{crs})$. It initializes a set I to the empty set.
  3. The challenger gives the attacker crs, com.
  4. The attacker can make an arbitrary number of queries of the form $i \in [k]$. When the attacker queries for a particular index i, the challenger responds with $\pi_i, \mathbf{r}_i$. The set I is updated to include the index i.
  5. If $b = 0$, the challenger gives $\mathbf{r}_{\bar{I}}$. If $b = 1$, the challenger chooses a random string $u \in \{0,1\}^{|\bar{I}|}$ and outputs u.

6. Algorithm $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$.

We say the Π_{HBG} is computationally hiding if for all efficient adversaries $\mathcal{A}$,

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

Theorem 2.2 (NIZK from Hidden-Bits Generator [KMY20, Theorem 5] (see also [FLS90, QRW19]))**.** *If there exists a hidden-bits generator according to Definition 2.1, then there exists a computational NIZK argument for* NP.

2.1 Single Bit Hiding

We now define the single bit hiding game for a hidden bits generator and show that it implies the adaptive computational hiding property from Definition 2.1.

Definition 2.3 (Single Bit Hiding)**.** For a security parameter λ and bit $b \in \{0,1\}$, we define the single bit computational hiding game between an adversary $\mathcal{A}$ and a challenger as follows:

1. On input the security parameter 1^λ, algorithm $\mathcal{A}$ starts by outputting the length parameter 1^k and an index $i^* \in [k]$.
2. The challenger samples $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^k)$, $(\mathsf{com}, \mathbf{r}, (\pi_1, \ldots, \pi_k)) \leftarrow \mathsf{GenBits}(\mathsf{crs})$.
3. The challenger gives the attacker crs, com as well as r_i, π_i for all $i \neq i^*$.
4. If $b = 0$, the challenger gives $\mathbf{r}_{i^*}$. If $b = 1$, the challenger chooses a random bit $u \in \{0,1\}$ and outputs u.
5. Algorithm $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$.

We say the Π_{HBG} is computationally single bit hiding if for all efficient adversaries $\mathcal{A}$,

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

Theorem 2.4 (Single bit hiding to adaptive hiding)**.** *If there exists a hidden-bits generator Π_{HBG} that satisfies single bit security of Definition 2.3, then Π_{HBG} satisfies the adaptive computational hiding of Definition 2.1.*

Proof. If there exists an attacker on the adaptive computational hiding game that wins with non-negligible probability, then for every security parameter λ, there exists an value $k^*(\lambda)$ such that the attacker wins with non-negligible probability while setting $k = k^*(\lambda)$. For simplicity we will assume an attacker that always makes this selection.

We define a set of hybrid experiments Hyb_j for $j \in [0,k]$ for each security parameter. The experiment runs the computational hiding experiment with the following exception. After Step 2 the challenger creates a string $\tilde{r} \in \{0,1\}^k$ where such that $\tilde{r}_{j'}$ is chosen randomly for $j' \leq j$. For $j' > j$ the experiment sets $\tilde{r}_{j'} = r_{j'}$. At the end of the experiment the attacker receives $\tilde{r}_{\bar{I}}$. We observe that Hyb_0 is equivalent to the computational hiding game when the bit $b = 0$ and that Hyb_k is equivalent to the computational hiding game when $b = 1$.

Claim 2.5. *Suppose Π_{HBG} is single bit hiding according to Definition 2.3, then for all $j \in [0, k-1]$ we have $|\Pr[\mathsf{Hyb}_j(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{j+1}(\mathcal{A}) = 1]| = \mathsf{negl}(\lambda)$.*

Proof. Suppose there exists an attacker $\mathcal{A}$ that has a non-negligible advantage is distinguishing between Hyb_j from Hyb_{j+1} (where j is some function of λ). Then we build an algorithm $\mathcal{B}$ that breaks the single bit hiding game.

- The algorithm $\mathcal{B}$ first submits to the challenger k^* and $i^* = j+1$.
- It receives back crs, com and $\mathbf{r}_i, \pi_i$ for all $i \neq j+1$. $\mathcal{B}$ then runs the attacker $\mathcal{A}$ giving the attacker crs, com.
- The attacker then makes queries for an index i. If $i \neq j+1$, the algorithm $\mathcal{B}$ gives $\mathbf{r}_i, \pi_i$ to $\mathcal{A}$. Otherwise if $i = j+1$ it aborts and makes a random guess b'.
- The algorithm $\mathcal{B}$ submits a guess δ, $\mathcal{B}$ outputs $b' = \delta$.

Suppose that $|\Pr[\mathsf{Hyb}_j(\mathcal{A}) = 1] - \Pr[\mathsf{Hyb}_{j+1}(\mathcal{A}) = 1]| = \epsilon(\lambda)$, then the advantage of $\mathcal{B} = \epsilon$. We first note that when the attacker queries on the index $j+1$ the distribution of the view of the $\mathcal{A}$ is the same in Hyb_j and Hyb_{j+1}. Therefore the difference between the probability that $\mathcal{A}$ outputs 1 in Hyb_j and does not query index $j+1$ and the probability that $\mathcal{A}$ outputs 1 in Hyb_{j+1} and does not query index $j+1$ is ϵ.

When $b = 0$ and no query on index $j+1$ is made the algorithm $\mathcal{A}$ sees experiment Hyb_j. When $b = 1$ and no query on index $j+1$ is made it has the view of experiment Hyb_{j+1}. The claim follows.

□

The theorem follows from (1) the above claim, (2) the fact that there are a polynomial $k+1$ number of hybrids and (3) the observation that Hyb_0 is equivalent to the computational hiding game with the bit $b = 0$ and that Hyb_k is equivalent to the computational hiding game with the bit $b = 1$.

□

Remark 2.6. We remark that our single point binding game is actually just a special case of the selective binding game considered in prior works. Thus our theorem actually shows that the selective hiding game from prior works implies the adaptive hiding game.

3 Learning with Errors and Trapdoors

Throughout this work, we use the ℓ_∞ norm for vectors and matrices. Specifically, for a vector $\mathbf{u}$, we write $\|\mathbf{u}\| := \max_i |x_i|$, and for a matrix $\mathbf{A}$, we write $\|\mathbf{A}\| = \max_{i,j} |A_{i,j}|$.

Discrete Gaussians. We write $D_{\mathbb{Z},\sigma}$ to denote the (centered) discrete Gaussian distribution over $\mathbb{Z}$ with parameter $\sigma \in \mathbb{R}^+$. For a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, and a vector $\mathbf{v} \in \mathbb{Z}_q^n$, we write $\mathbf{A}_\sigma^{-1}(\mathbf{v})$ to denote a random variable $\mathbf{x} \leftarrow D_{\mathbb{Z},\chi}^m$ conditioned on $\mathbf{Ax} = \mathbf{v} \bmod q$. We extend $\mathbf{A}_s^{-1}$ to matrices by applying $\mathbf{A}_s^{-1}$ to each column of the input. Throughout this work, we will use the following standard tail bound on Gaussian distributions:

Fact 3.1 (Gaussian Tail Bound)**.** Let λ be a security parameter and $\sigma = \sigma(\lambda)$ be a Gaussian width parameter. Then, for all polynomials $n = n(\lambda)$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that for all $\lambda \in \mathbb{N}$,

$$\Pr\left[\, \|\mathbf{v}\|_\infty > \sqrt{\lambda}\sigma : \mathbf{v} \leftarrow D_{\mathbb{Z},\sigma}^m \right] = \mathsf{negl}(\lambda).$$

Truncated Discrete Gaussians We write $\tilde{D}_{\mathbb{Z},\sigma}$ to denote a truncated Discrete Gaussian distribution. This distribution will be the same as $D_{\mathbb{Z},\sigma}$ except that it outputs the zero vector $\mathbf{0}$ if the infinity norm of the output ever exceeds $\sqrt{\lambda}\sigma$. By definition the infinity norm of this distribution is bounded by $\sqrt{\lambda}\sigma$ and by Fact 3.1 the distributions $\tilde{D}_{\mathbb{Z},\sigma}$ and $D_{\mathbb{Z},\sigma}$ are statistically close.

Lemma 3.2 (Smudging Lemma [AJL+12, Lemma 2.1, paraphrased])**.** *Let B_1, B_2 be two polynomials over the integers and let $D = \{D(\lambda)\}_\lambda$ be any B_1-bounded distribution family. Let $U = \{U(\lambda)\}_\lambda$ and $U(\lambda)$ denote the uniform distribution over integers $[-B_2(\lambda), B_2(\lambda)]$. The family of distributions D and U is statistically indistinguishable, $D + U \approx_s U$, if there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $B_1(\lambda)/B_2(\lambda) \le \mathsf{negl}(\lambda)$.*

Assumption 3.3 (Learning with Errors [Reg05])**.** Let λ be a security parameter and let $n = n(\lambda)$, $m = m(\lambda)$, $q = q(\lambda)$ be integers and $\sigma = \sigma(\lambda)$ be a Gaussian width parameter. Then, the decisional learning with errors assumption $\mathsf{LWE}_{n,m,q,\sigma}$ states that for $\mathbf{A} \xleftarrow{R} \mathbb{Z}_q^{n\times m}$, $\mathbf{s} \xleftarrow{R} \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow D_{\mathbb{Z},\sigma}^m$, and $\mathbf{u} \xleftarrow{R} \mathbb{Z}_q^m$,

$$(\mathbf{A}, \mathbf{s}^\intercal\mathbf{A} + \mathbf{e}^\intercal) \overset{c}{\approx} (\mathbf{A}, \mathbf{u}).$$

Theorem 3.4 (Lattice Trapdoors [Ajt96, GPV08, ABB10b, ABB10a, CHKP10, MP12])**.** *Let n, m, q be lattice parameters. Then there exist efficient algorithms* (TrapGen, SamplePre) *with the following syntax:*

- $\mathsf{TrapGen}(1^n, q, m) \to (\mathbf{A}, \mathsf{td}_\mathbf{A})$*: On input the lattice dimension n, the modulus q, the number of samples m, the trapdoor-generation algorithm outputs a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ together with a trapdoor $\mathsf{td}_\mathbf{A}$.*

- $\mathsf{SamplePre}(\mathbf{A}, \mathsf{td}_{\mathbf{A}}, \mathbf{v}, s) \to \mathbf{u}$*: On input a matrix* $\mathbf{A}$*, a trapdoor* $\mathsf{td}_{\mathbf{A}}$*, a target vector* $\mathbf{v}$*, and a Gaussian width parameter* s*, the preimage-sampling algorithm outputs a vector* $\mathbf{u}$*.*

Moreover, there exists a polynomial $m_0 = m_0(n, q) = O(n \log q)$ *such that for all* $m \geq m_0$*, the above algorithms satisfy the following properties:*

- **Trapdoor distribution:** *The matrix* $\mathbf{A}$ *output by* $\mathsf{TrapGen}(1^n, q, m)$ *is statistically close to uniform. Specifically, if* $(\mathbf{A}, \mathsf{td}_{\mathbf{A}}) \leftarrow \mathsf{TrapGen}(1^n, q, m)$ *and* $\mathbf{A}' \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}$*, then* $\Delta(\mathbf{A}, \mathbf{A}') \leq 2^{-n}$*.*
- **Trapdoor quality:** *The trapdoor* $\mathsf{td}_{\mathbf{A}}$ *output by* $\mathsf{TrapGen}(1^n, q, m)$ *is a* τ*-trapdoor where* $\tau = O\left(\sqrt{n \log q \log n}\right)$*. We refer to the parameter* τ *as the* quality *of the trapdoor.*
- **Preimage sampling:** *Suppose* $\mathsf{td}_{\mathbf{A}}$ *is a* τ*-trapdoor for* $\mathbf{A}$*. Then, for all* $s \geq \tau \cdot \omega(\sqrt{\log n})$ *and all target vectors* $\mathbf{v} \in \mathbb{Z}_q^n$*, the statistical distance between the following distributions is at most* 2^{-n}*:*

$$\{\mathbf{u} \leftarrow \mathsf{SamplePre}(\mathbf{A}, \mathsf{td}_{\mathbf{A}}, \mathbf{v}, s)\} \quad \textit{and} \quad \left\{\mathbf{u} \leftarrow \mathbf{A}_s^{-1}(\mathbf{v})\right\}.$$

Preimage sampling for random targets. We will also use the following property of discrete Gaussian distributions which follows from [GPV08]:

Lemma 3.5 (Preimage Sampling of random targets [GPV08, adapted])**.** *Let* n, m, q *be lattice parameters. There exists polynomials* $m_0(n, q) = O(n \log q)$ *and* $\sigma_0(n, q) = \sqrt{n \log q} \cdot \omega(\sqrt{\log n})$ *such that for all* $m \geq m_0(n, q)$ *and* $\sigma \geq \sigma_0(n, q)$*, the statistical distance between the following distributions is* $\mathsf{negl}(n)$*:*

$$\left\{(\mathbf{A}, \mathbf{x}, \mathbf{Ax}) : \mathbf{A} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}, \mathbf{x} \leftarrow D_{\mathbb{Z},\sigma}^m\right\} \textit{ and } \left\{(\mathbf{A}, \mathbf{x}, \mathbf{y}) : \mathbf{A} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n \times m}, \mathbf{y} \xleftarrow{\mathrm{R}} \mathbb{Z}_q^n, \mathbf{x} \leftarrow \mathbf{A}_\sigma^{-1}(\mathbf{y})\right\}.$$

In our proofs of security we will sometimes use a game-based description of the preimage sampling and trapdoor distribution properties where the attacker receives a sample and is required to distinguish which distribution it came from.

4 Our Hidden Bits Generator Construction

Parameterization Prior to describing our construction we describe how we will set the parameters used in it. This is particularly important in our construction since the proof of security will be strongly connected with the choice of parameters. We prioritize on the ease of exposition and verification of our results by providing a specific setting of the parameters. We note that there will exist other interesting choices of parameters beyond what is given below.

All parameters below will depend on at least one of (1) the security parameter λ, (2) the length parameter k given by the attacker and (3) a constant $\gamma \in (0, \frac{1}{2})$. We set the parameters as follows:

- Modulus q: Chosen to be a prime where $2^\lambda < q < 2^{\lambda+1}$
- n: $\lambda^{1/\gamma}$
- σ: $n^{1.1}$
- m: $2(n+1)\lg(q)$
- L: $\lambda \cdot m \cdot k + 5\lambda + 1$
- B: $2^{.5\lambda}$
- $\mathtt{TestBound}$: $\sigma\sqrt{\lambda} \cdot L \cdot B$
- $\mathtt{RoundingBound}$: $\sigma\sqrt{\lambda} \cdot m \cdot \mathtt{TestBound}$

Construction 4.1. We now describe the algorithms of our hidden bits generator Π_{HBG}.

- $\mathsf{Setup}(1^\lambda, 1^k) \to \mathsf{crs}$:
  1. Run $(\mathbf{A}_i, \mathsf{td}_i) \leftarrow \mathsf{TrapGen}(1^n, q, m)$ for all $i \in [k]$.
  2. Choose random $\mathbf{U} \xleftarrow{R} \mathbb{Z}_q^{n \times L}$.
  3. Sample
     (a) $\mathbf{s}_i \xleftarrow{R} \mathbb{Z}_q^n$ for all $i \in [k]$
     (b) $\mathbf{e}_i \xleftarrow{R} \tilde{D}_{\mathbb{Z},\sigma}^m$ for all $i \in [k]$
     (c) $d_i \xleftarrow{R} \mathbb{Z}_q$ for all $i \in [k]$
  4. Compute $\mathbf{v}_i^\intercal = \mathbf{s}_i^\intercal \mathbf{A}_i + \mathbf{e}_i^\intercal$ for all $i \in [k]$.
  5. Sample $\mathbf{W}_i \xleftarrow{R} \mathsf{SamplePre}(\mathbf{A}_i, \mathsf{td}_i, \mathbf{U}, \sigma)$ for all $i \in [k]$.
  6. Output $\mathsf{crs} = \{\mathbf{U}, \mathbf{A}_i, \mathbf{W}_i, \mathbf{v}_i, d_i\}_{i \in [k]}$.
- $\mathsf{GenBits}(\mathsf{crs}) \to (\mathsf{com}, \mathbf{r}, (\pi_1, \ldots, \pi_k))$:
  1. Sample $\mathbf{t} \xleftarrow{R} [-B, B]^L$.
  2. Compute $\pi_i = \mathbf{W}_i \mathbf{t}$ for all $i \in [k]$.
  3. Set $r_i = \lfloor \mathbf{v}_i^\intercal \pi_i + d_i \rceil$ for all $i \in [k]$.
  4. Set $\mathsf{com} = \mathbf{U}\mathbf{t}$.
  5. Output $(\mathsf{com}, \mathbf{r}, (\pi_1, \ldots, \pi_k))$.
- $\mathsf{Verify}(\mathsf{crs}, \mathsf{com}, i, \beta, \pi) \to b$:
  1. Check if $\|\pi\|_\infty \le \mathtt{TestBound}$.
  2. Check if $\mathsf{com} \stackrel{?}{=} \mathbf{A}_i \pi$.
  3. Check if $\beta \stackrel{?}{=} \lfloor \mathbf{v}_i^\intercal \pi + d_i \rceil$.
  4. Check if $\beta \stackrel{?}{=} \lfloor \mathbf{v}_i^\intercal \pi + d_i + \mathtt{RoundingBound} \rceil$ and $\beta \stackrel{?}{=} \lfloor \mathbf{v}_i^\intercal \pi + d_i - \mathtt{RoundingBound} \rceil$.
  5. Output 1 if all tests hold and 0 otherwise.

Remark 4.2. While our construction passes crs to both the GenBits and Verify algorithms, we note that we could separate out a significantly shorter verification crs by including just $\{\mathbf{A}_i, \mathbf{v}_i, d_i\}_{in \in [k]}$ since only these values are used in verification. The crs of the prover needs to grow with k^2, but the a separated out verifier key could grow linearly in k. Finally, we could use signatures to push most of that shorter verification key into the proof.

Correctness

Theorem 4.3. *Construction 4.1 satisfies correctness of Definition 2.1 with all but negligible probability over the choice of the coins of* Setup *and* GenBits.

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

The trapdoor produced from setting $(\mathbf{A}_i, \mathsf{td}_i) \leftarrow \mathsf{TrapGen}(1^n, q, m)$ are of quality $\tau = O(\sqrt{n \log q \log n}) = O(\sqrt{n\lambda \log n})$. Since $\sigma \ge \tau \cdot \omega(\sqrt{\log n})$ the entries of $\mathbf{W}_i$ for all i are at most $\sigma\sqrt{\lambda}$.

In computing π_i we do matrix multiplication of $\mathbf{W}_i$ with $\mathbf{t}$ which is an L-length vector with entries in $[-B, B]$. Thus the result will be a vector with entries whose absolute value is at most $\sigma\sqrt{\lambda}LB$ which matches $\mathtt{TestBound}$.

For the second check we observe:

$$\begin{aligned} \mathbf{A}_i \pi_i &= \mathbf{A}_i \mathbf{W}_i \mathbf{t} && \text{By computation of } \pi_i \\ &= \mathbf{U}\mathbf{t} && \text{Since } \mathbf{W}_i \xleftarrow{R} \mathsf{SamplePre}(\mathbf{A}_i, \mathsf{td}_i, \mathbf{U}, \sigma) \\ &= \mathsf{com} \end{aligned}$$

Next, we observe that the verification equation tests if $\beta \stackrel{?}{=} \lfloor \mathbf{v}^\intercal \pi + d_i \rceil$. However, the GenBits algorithm computed r_i from π_i in exactly the same way. Thus if they were set correctly this check will also pass.

Finally, the verification algorithm checks to see if the output answer would flip if either the value RoundingBound were added to or subtracted from $\mathbf{v}_i^\intercal \pi + d_i$. If so, then the algorithm rejects. This final check will cause correctness error with at most negligible probability. For the round function there exists a single value $a_0 \in \mathbb{Z}_q$ such that $\lfloor a_0 \rceil = 0$, but $\lfloor a_0 + 1 \rceil = 1$. And a different value a_1 such that $\lfloor a_1 \rceil = 1$, but $\lfloor a_1 + 1 \rceil = 0$. The tests will fail if either a_0 or a_1 is in the range $[\mathbf{v}^\intercal \pi + d_i - \texttt{RoundingBound}, \mathbf{v}^\intercal \pi + d_i + \texttt{RoundingBound} - 1]$. The probability of either of these occurring is $4 \cdot \texttt{RoundingBound}/q$ due to the fact that d_i is chosen uniformly in $\mathbb{Z}_q$ and independent of all other values from Setup and GenBits. Since RoundingBound is $2^{\cdot 5\lambda} \cdot \text{poly}(\lambda)$ we have that $4 \cdot \texttt{RoundingBound}/q$ is negligible in λ as needed. □

4.1 Moving to Perfect Correctness Generically

We briefly sketch a generic method to transform a hidden bits generator with negligible correctness error to one with perfect correctness.

In the new system the Setup algorithm will remain the same. The GenBits algorithm will be modified as follows. (1) First, run the old GenBits algorithm to get $(\mathsf{com}, \mathbf{r}, (\pi_1, \ldots, \pi_k))$. (2) For all $i \in [k]$ run $\mathsf{Verify}(\mathsf{crs}, \mathsf{com}, i, \mathbf{r}_i, \pi_i)$. If all checks pass, use $(\mathsf{com}, \mathbf{r}, (\pi_1, \ldots, \pi_k))$ from above. (3) Otherwise, set com to be a special $\perp$ symbol, set $\mathbf{r} = 0^k$ (the all 0s string) and let π_i be empty for all $i \in [k]$. Essentially, any correctness errors in the first commitment trigger a special commitment to the all 0s string. The new $\mathsf{Verify}(\mathsf{crs}, \mathsf{com}, i, \beta, \pi)$ will behave exactly as before if $\mathsf{com} \neq \perp$. Otherwise, it accepts if and only $\beta = 0$.

The new system is perfectly correct. The GenBits algorithm tests if there could be any correctness errors in its originally sampled $(\mathsf{com}, \mathbf{r}, (\pi_1, \ldots, \pi_k))$ values. If not, we are fine. If there are any, it moves to the special $\mathsf{com} = \perp$, $\mathbf{r} = 0^k$ values which are guaranteed to verify. The transformation will not impact binding security for the original algorithm and the $\mathsf{com} = \perp$ case can only be opened to 0 values. This adds at most one string to $\mathcal{V}^{\mathsf{crs}}$. Finally, hiding security can only be impacted if the special $\mathsf{com} = \perp$ condition is triggered. However, there was a negligible correctness error to begin with and this will only happen with negligible probability for an honest execution of the Setup and GenBits algorithms.

5 Proof of Binding Security

Theorem 5.1. *Our construction achieves statistical binding per Definition 2.1 with the output sparsity parameter* $|\mathcal{V}^{\mathsf{crs}}| \leq 2^{(\lambda+1)n}$.

Proof. Each commitment is a tuple in $\mathbb{Z}_q^n$ where q is a prime at most $2^{\lambda+1}$. The number of possible commitments is therefore at most $2^{(\lambda+1)n}$. We now just need to show that for a particular crs, commitment com and index value $i \in [k]$ can be opened to at most a single bit value.

We show that either $\left\lfloor \mathbf{v}_i^\intercal \pi + d_i \right\rceil$ must take on the single "canoncial" value $\left\lfloor \mathbf{s}_i^\intercal \mathsf{com} + d_i \right\rceil$ or it will be rejected by the final check in the verify algorithm of Construction 4.1.

Consider a proof π such that $\mathsf{Verify}(\mathsf{crs}, \mathsf{com}, i, \beta, \pi) \to 1$ for some β. Since the proof verifies we have that $\mathsf{com} = \mathbf{A}_i \pi$. It follows that

$$\left\lfloor \mathbf{v}_i^\intercal \pi + d_i \right\rceil = \left\lfloor (\mathbf{s}_i^\intercal \mathbf{A}_i + \mathbf{e}_i^\intercal)\pi + d_i \right\rceil = \left\lfloor \mathbf{s}_i^\intercal \mathsf{com} + \mathbf{e}_i^\intercal \pi + d_i \right\rceil.$$

Next we bound the value that $\mathbf{e}_i^\intercal \pi$ can take. Each entry of $\mathbf{e}_i$ produced from setup is at most $\sqrt{\lambda}\sigma$ and since verification passed each entry of π is at most TestBound. Since the vectors are of length m the most $\mathbf{e}_i^\intercal \pi$ can be is $\texttt{TestBound} \cdot \sqrt{\lambda}\sigma m = \texttt{RoundingBound}$. Likewise, the least value of $\mathbf{e}_i^\intercal \pi$ is $-\texttt{TestBound} \cdot \sqrt{\lambda}\sigma m = -\texttt{RoundingBound}$. This means that $\mathbf{s}_i^\intercal \mathsf{com} + \mathbf{e}_i^\intercal \pi + d_i$ must be within RoundingBound of $\mathbf{s}_i^\intercal \mathsf{com} + d_i$. Therefore if $\left\lfloor \mathbf{s}_i^\intercal \mathsf{com} + d_i \right\rceil \neq \left\lfloor \mathbf{s}_i^\intercal \mathsf{com} + \mathbf{e}_i^\intercal \pi + d_i \right\rceil$, one of the checks of the final verification step will also not be equal and the algorithm will reject. □

6 Mode Indistinguishability

We now prove that our construction is single bit hiding per Definition 2.3. The first step to our proof is to introduce a lossy setup algorithm SetupHiding. In this mode the output of any bit from GenBits will be statistically hidden even given all other proofs.

Our system therefore has a dual mode type property where it can be setup to be either statistically binding or statistically hiding. We describe the hiding setup algorithm below and show that its output is computationally indistinguishable under the LWE assumption from that of the normal setup.

$\mathsf{SetupHiding}(1^\lambda, 1^k) \rightarrow \mathsf{crs}$:

1. Run $(\mathbf{A}'_i, \mathsf{td}_i) \leftarrow \mathsf{TrapGen}(1^{n+1}, q, m)$ and parse $\mathbf{A}'_i = \begin{bmatrix} \mathbf{A}_i \\ \mathbf{v}_i^\top \end{bmatrix}$ where $\mathbf{A}_i \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{v}_i$ is a vector for all $i \in [k]$.
2. Choose random $\mathbf{U} \xleftarrow{R} \mathbb{Z}_q^{n \times L}$.
3. Sample
   (a) $\mathbf{u}_i \xleftarrow{R} \mathbb{Z}_q^L$ for all $i \in [k]$
   (b) $d_i \xleftarrow{R} \mathbb{Z}_q$ for all $i \in [k]$
4. Set $\mathbf{U}'_i = \begin{bmatrix} \mathbf{U} \\ \mathbf{u}_i^\top \end{bmatrix}$ for all $i \in [k]$.
5. Sample $\mathbf{W}_i \xleftarrow{R} \mathsf{SamplePre}(\mathbf{A}'_i, \mathsf{td}_i, \mathbf{U}'_i, \sigma)$ for all $i \in [k]$.
6. Output $\mathsf{crs} = \{\mathbf{U}, \mathbf{A}_i, \mathbf{W}_i, \mathbf{v}_i, d_i\}_{i \in [k]}$.

6.1 Proof of Indistinguishability

We define a sequence of games where the first is Definition 2.3 applied to our construction. We will use these to gradually change the distribution of the setup algorithm in the sequence until the game is using the alternative setup. Our game definitions will be double scripted of the form (i^*, j) where $i^* \in [k]$ and $j \in [1, 7]$. For clarity of exposition we choose to write out the modified setup algorithm in full for each game definition where we highlight the changes in color. While this presentation will use more page space, it makes it easier for the reader to keep track of the status when we are multiple games into the proof.

- $\mathsf{Game}_{i^*,1}$: Initial form where parameters are set in one fashion for all $i < i^*$ and another for $i \geq i^*$.
  1. Run $(\mathbf{A}_i, \mathsf{td}_i) \leftarrow \mathsf{TrapGen}(1^n, q, m)$ for all $i \in [i^*, k]$.
  2. Run $(\mathbf{A}'_i, \mathsf{td}_i) \leftarrow \mathsf{TrapGen}(1^{n+1}, q, m)$ and parse $\mathbf{A}'_i = \begin{bmatrix} \mathbf{A}_i \\ \mathbf{v}_i^\top \end{bmatrix}$ where $\mathbf{A}_i \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{v}_i$ is a vector for all $i \in [1, i^* - 1]$.
  3. Choose random $\mathbf{U} \xleftarrow{R} \mathbb{Z}_q^{n \times L}$.
  4. Sample
     (a) $\mathbf{s}_i \xleftarrow{R} \mathbb{Z}_q^n$ for all $i \in [i^*, k]$
     (b) $\mathbf{e}_i \xleftarrow{R} \tilde{D}_{\mathbb{Z},\sigma}^m$ for all $i \in [i^*, k]$
     (c) $\mathbf{u}_i \xleftarrow{R} \mathbb{Z}_q^L$ for all $i \in [1, i^* - 1]$
     (d) $d_i \xleftarrow{R} \mathbb{Z}_q$ for all $i \in [k]$
  5. Compute $\mathbf{v}_i^\top = \mathbf{s}_i^\top \mathbf{A}_i + \mathbf{e}_i^\top$ for all $i \in [i^*, k]$.
  6. Set $\mathbf{U}'_i = \begin{bmatrix} \mathbf{U} \\ \mathbf{u}_i^\top \end{bmatrix}$ for all $i \in [1, i^* - 1]$.

7. Sample $\mathbf{W}_i \xleftarrow{R} \mathsf{SamplePre}(\mathbf{A}_i, \mathsf{td}_i, \mathbf{U}, \sigma)$ for all $i \in [i^*, k]$.
8. Sample $\mathbf{W}_i \xleftarrow{R} \mathsf{SamplePre}(\mathbf{A}'_i, \mathsf{td}_i, \mathbf{U}'_i, \sigma)$ for all $i \in [1, i^* - 1]$.
9. Output $\mathsf{crs} = \{\mathbf{U}, \mathbf{A}_i, \mathbf{W}_i, \mathbf{v}_i, d_i\}_{i \in [k]}$.

- $\mathsf{Game}_{i^*,2}$: Same as $\mathsf{Game}_{i^*,1}$ except the setup algorithm is run as:

1. Run $(\mathbf{A}_i, \mathsf{td}_i) \leftarrow \mathsf{TrapGen}(1^n, q, m)$ for all $i \in [i^*, k]$.
2. Run $(\mathbf{A}'_i, \mathsf{td}_i) \leftarrow \mathsf{TrapGen}(1^{n+1}, q, m)$ and parse $\mathbf{A}'_i = \begin{bmatrix} \mathbf{A}_i \\ \mathbf{v}_i^\top \end{bmatrix}$ where $\mathbf{A}_i \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{v}_i$ is a vector for all $i \in [1, i^* - 1]$.
3. $\mathbf{W}_{i^*} \xleftarrow{R} D_{\mathbb{Z},\sigma}^{m \times L}$.
4. $\mathbf{U} = \mathbf{A}_{i^*} \mathbf{W}_{i^*}$.
5. Sample
   (a) $\mathbf{s}_i \xleftarrow{R} \mathbb{Z}_q^n$ for all $i \in [i^*, k]$
   (b) $\mathbf{e}_i \xleftarrow{R} \tilde{D}_{\mathbb{Z},\sigma}^m$ for all $i \in [i^*, k]$
   (c) $\mathbf{u}_i \xleftarrow{R} \mathbb{Z}_q^L$ for all $i \in [1, i^* - 1]$
   (d) $d_i \xleftarrow{R} \mathbb{Z}_q$ for all $i \in [k]$
6. Compute $\mathbf{v}_i^\top = \mathbf{s}_i^\top \mathbf{A}_i + \mathbf{e}_i^\top$ for all $i \in [i^*, k]$.
7. Set $\mathbf{U}'_i = \begin{bmatrix} \mathbf{U} \\ \mathbf{u}_i^\top \end{bmatrix}$ for all $i \in [1, i^* - 1]$.
8. Sample $\mathbf{W}_i \xleftarrow{R} \mathsf{SamplePre}(\mathbf{A}_i, \mathsf{td}_i, \mathbf{U}, \sigma)$ for all $i \in [i^* + 1, k]$.
9. Sample $\mathbf{W}_i \xleftarrow{R} \mathsf{SamplePre}(\mathbf{A}'_i, \mathsf{td}_i, \mathbf{U}'_i, \sigma)$ for all $i \in [1, i^* - 1]$.
10. Output $\mathsf{crs} = \{\mathbf{U}, \mathbf{A}_i, \mathbf{W}_i, \mathbf{v}_i, d_i\}_{i \in [k]}$.

- $\mathsf{Game}_{i^*,3}$: Same as $\mathsf{Game}_{i^*,2}$ except the setup algorithm is run as:

1. Run $(\mathbf{A}_i, \mathsf{td}_i) \leftarrow \mathsf{TrapGen}(1^n, q, m)$ for all $i \in [i^* + 1, k]$.
2. Run $(\mathbf{A}'_i, \mathsf{td}_i) \leftarrow \mathsf{TrapGen}(1^{n+1}, q, m)$ and parse $\mathbf{A}'_i = \begin{bmatrix} \mathbf{A}_i \\ \mathbf{v}_i^\top \end{bmatrix}$ where $\mathbf{A}_i \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{v}_i$ is a vector for all $i \in [1, i^* - 1]$.
3. $\mathbf{A}_{i^*} \xleftarrow{R} \mathbb{Z}_q^{n \times m}$.
4. $\mathbf{W}_{i^*} \xleftarrow{R} D_{\mathbb{Z},\sigma}^{m \times L}$.
5. $\mathbf{U} = \mathbf{A}_{i^*} \mathbf{W}_{i^*}$.
6. Sample
   (a) $\mathbf{s}_i \xleftarrow{R} \mathbb{Z}_q^n$ for all $i \in [i^*, k]$
   (b) $\mathbf{e}_i \xleftarrow{R} \tilde{D}_{\mathbb{Z},\sigma}^m$ for all $i \in [i^*, k]$
   (c) $\mathbf{u}_i \xleftarrow{R} \mathbb{Z}_q^L$ for all $i \in [1, i^* - 1]$
   (d) $d_i \xleftarrow{R} \mathbb{Z}_q$ for all $i \in [k]$
7. Compute $\mathbf{v}_i^\top = \mathbf{s}_i^\top \mathbf{A}_i + \mathbf{e}_i^\top$ for all $i \in [i^*, k]$.
8. Set $\mathbf{U}'_i = \begin{bmatrix} \mathbf{U} \\ \mathbf{u}_i^\top \end{bmatrix}$ for all $i \in [1, i^* - 1]$.
9. Sample $\mathbf{W}_i \xleftarrow{R} \mathsf{SamplePre}(\mathbf{A}_i, \mathsf{td}_i, \mathbf{U}, \sigma)$ for all $i \in [i^* + 1, k]$.
10. Sample $\mathbf{W}_i \xleftarrow{R} \mathsf{SamplePre}(\mathbf{A}'_i, \mathsf{td}_i, \mathbf{U}'_i, \sigma)$ for all $i \in [1, i^* - 1]$.

11. Output crs $= \{\mathbf{U}, \mathbf{A}_i, \mathbf{W}_i, \mathbf{v}_i, d_i\}_{i\in[k]}$.

- $\mathsf{Game}_{i^*,4}$: Same as $\mathsf{Game}_{i^*,3}$ except the setup algorithm is run as:
  1. Run $(\mathbf{A}_i, \mathsf{td}_i) \leftarrow \mathsf{TrapGen}(1^n, q, m)$ for all $i \in [i^*+1, k]$.
  2. Run $(\mathbf{A}'_i, \mathsf{td}_i) \leftarrow \mathsf{TrapGen}(1^{n+1}, q, m)$ and parse $\mathbf{A}'_i = \begin{bmatrix} \mathbf{A}_i \\ \mathbf{v}_i^\intercal \end{bmatrix}$ where $\mathbf{A}_i \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{v}_i$ is a vector for all $i \in [1, i^*-1]$.
  3. $\mathbf{A}_{i^*} \xleftarrow{R} \mathbb{Z}_q^{n\times m}$ and $\mathbf{v}_{i^*} \xleftarrow{R} \mathbb{Z}_q^m$. Let $\mathbf{A}'_{i^*} = \begin{bmatrix} \mathbf{A}_{i^*} \\ \mathbf{v}_{i^*}^\intercal \end{bmatrix}$.
  4. $\mathbf{W}_{i^*} \xleftarrow{R} D_{\mathbb{Z},\sigma}^{m\times L}$.
  5. $\begin{bmatrix} \mathbf{U} \\ \mathbf{u}_{i^*}^\intercal \end{bmatrix} = \mathbf{A}'_{i^*}\mathbf{W}_{i^*}$.
  6. Sample
     (a) $\mathbf{s}_i \xleftarrow{R} \mathbb{Z}_q^n$ for all $i \in [i^*+1, k]$
     (b) $\mathbf{e}_i \xleftarrow{R} \tilde{D}_{\mathbb{Z},\sigma}^m$ for all $i \in [i^*+1, k]$
     (c) $\mathbf{u}_i \xleftarrow{R} \mathbb{Z}_q^L$ for all $i \in [1, i^*-1]$
     (d) $d_i \xleftarrow{R} \mathbb{Z}_q$ for all $i \in [k]$
  7. Compute $\mathbf{v}_i^\intercal = \mathbf{s}_i^\intercal \mathbf{A}_i + \mathbf{e}_i^\intercal$ for all $i \in [i^*+1, k]$.
  8. Set $\mathbf{U}'_i = \begin{bmatrix} \mathbf{U} \\ \mathbf{u}_i^\intercal \end{bmatrix}$ for all $i \in [1, i^*-1]$.
  9. Sample $\mathbf{W}_i \xleftarrow{R} \mathsf{SamplePre}(\mathbf{A}_i, \mathsf{td}_i, \mathbf{U}, \sigma)$ for all $i \in [i^*+1, k]$.
  10. Sample $\mathbf{W}_i \xleftarrow{R} \mathsf{SamplePre}(\mathbf{A}'_i, \mathsf{td}_i, \mathbf{U}'_i, \sigma)$ for all $i \in [1, i^*-1]$.
  11. Output crs $= \{\mathbf{U}, \mathbf{A}_i, \mathbf{W}_i, \mathbf{v}_i, d_i\}_{i\in[k]}$.

- $\mathsf{Game}_{i^*,5}$: Same as $\mathsf{Game}_{i^*,4}$ except the setup algorithm is run as:
  1. Run $(\mathbf{A}_i, \mathsf{td}_i) \leftarrow \mathsf{TrapGen}(1^n, q, m)$ for all $i \in [i^*+1, k]$.
  2. Run $(\mathbf{A}'_i, \mathsf{td}_i) \leftarrow \mathsf{TrapGen}(1^{n+1}, q, m)$ and parse $\mathbf{A}'_i = \begin{bmatrix} \mathbf{A}_i \\ \mathbf{v}_i^\intercal \end{bmatrix}$ where $\mathbf{A}_i \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{v}_i$ is a vector for all $i \in [1, i^*]$.
  3. $\mathbf{W}_{i^*} \xleftarrow{R} D_{\mathbb{Z},\sigma}^{m\times L}$.
  4. $\begin{bmatrix} \mathbf{U} \\ \mathbf{u}_{i^*}^\intercal \end{bmatrix} = \mathbf{A}'_{i^*}\mathbf{W}_{i^*}$.
  5. Sample
     (a) $\mathbf{s}_i \xleftarrow{R} \mathbb{Z}_q^n$ for all $i \in [i^*+1, k]$
     (b) $\mathbf{e}_i \xleftarrow{R} \tilde{D}_{\mathbb{Z},\sigma}^m$ for all $i \in [i^*+1, k]$
     (c) $\mathbf{u}_i \xleftarrow{R} \mathbb{Z}_q^L$ for all $i \in [1, i^*-1]$
     (d) $d_i \xleftarrow{R} \mathbb{Z}_q$ for all $i \in [k]$
  6. Compute $\mathbf{v}_i^\intercal = \mathbf{s}_i^\intercal \mathbf{A}_i + \mathbf{e}_i^\intercal$ for all $i \in [i^*+1, k]$.
  7. Set $\mathbf{U}'_i = \begin{bmatrix} \mathbf{U} \\ \mathbf{u}_i^\intercal \end{bmatrix}$ for all $i \in [1, i^*-1]$.
  8. Sample $\mathbf{W}_i \xleftarrow{R} \mathsf{SamplePre}(\mathbf{A}_i, \mathsf{td}_i, \mathbf{U}, \sigma)$ for all $i \in [i^*+1, k]$.
  9. Sample $\mathbf{W}_i \xleftarrow{R} \mathsf{SamplePre}(\mathbf{A}'_i, \mathsf{td}_i, \mathbf{U}'_i, \sigma)$ for all $i \in [1, i^*-1]$.

10. Output crs $= \{\mathbf{U}, \mathbf{A}_i, \mathbf{W}_i, \mathbf{v}_i, d_i\}_{i\in[k]}$.

- $\mathsf{Game}_{i^*,6}$: Same as $\mathsf{Game}_{i^*,5}$ except the setup algorithm is run as:
  1. Run $(\mathbf{A}_i, \mathsf{td}_i) \leftarrow \mathsf{TrapGen}(1^n, q, m)$ for all $i \in [i^*+1, k]$.
  2. Run $(\mathbf{A}'_i, \mathsf{td}_i) \leftarrow \mathsf{TrapGen}(1^{n+1}, q, m)$ and parse $\mathbf{A}'_i = \begin{bmatrix} \mathbf{A}_i \\ \mathbf{v}_i^\intercal \end{bmatrix}$ where $\mathbf{A}_i \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{v}_i$ is a vector for all $i \in [1, i^*]$.
  3. Choose random $\begin{bmatrix} \mathbf{U} \\ \mathbf{u}_{i^*}^\intercal \end{bmatrix} \xleftarrow{R} \mathbb{Z}_q^{n+1\times L}$.
  4. Sample $\mathbf{W}_{i^*} \xleftarrow{R} \mathsf{SamplePre}(\mathbf{A}'_{i^*}, \mathsf{td}_{i^*}, \mathbf{U}'_{i^*}, \sigma)$.
  5. Sample
     (a) $\mathbf{s}_i \xleftarrow{R} \mathbb{Z}_q^n$ for all $i \in [i^*+1, k]$
     (b) $\mathbf{e}_i \xleftarrow{R} \tilde{D}_{\mathbb{Z},\sigma}^m$ for all $i \in [i^*+1, k]$
     (c) $\mathbf{u}_i \xleftarrow{R} \mathbb{Z}_q^L$ for all $i \in [1, i^*-1]$
     (d) $d_i \xleftarrow{R} \mathbb{Z}_q$ for all $i \in [k]$
  6. Compute $\mathbf{v}_i^\intercal = \mathbf{s}_i^\intercal \mathbf{A}_i + \mathbf{e}_i^\intercal$ for all $i \in [i^*+1, k]$.
  7. Set $\mathbf{U}'_i = \begin{bmatrix} \mathbf{U} \\ \mathbf{u}_i^\intercal \end{bmatrix}$ for all $i \in [1, i^*-1]$.
  8. Sample $\mathbf{W}_i \xleftarrow{R} \mathsf{SamplePre}(\mathbf{A}_i, \mathsf{td}_i, \mathbf{U}, \sigma)$ for all $i \in [i^*+1, k]$.
  9. Sample $\mathbf{W}_i \xleftarrow{R} \mathsf{SamplePre}(\mathbf{A}'_i, \mathsf{td}_i, \mathbf{U}'_i, \sigma)$ for all $i \in [1, i^*-1]$.
  10. Output crs $= \{\mathbf{U}, \mathbf{A}_i, \mathbf{W}_i, \mathbf{v}_i, d_i\}_{i\in[k]}$.
- $\mathsf{Game}_{i^*,7}$: Same as $\mathsf{Game}_{i^*,6}$ except the setup algorithm is run as:
  1. Run $(\mathbf{A}_i, \mathsf{td}_i) \leftarrow \mathsf{TrapGen}(1^n, q, m)$ for all $i \in [i^*+1, k]$.
  2. Run $(\mathbf{A}'_i, \mathsf{td}_i) \leftarrow \mathsf{TrapGen}(1^{n+1}, q, m)$ and parse $\mathbf{A}'_i = \begin{bmatrix} \mathbf{A}_i \\ \mathbf{v}_i^\intercal \end{bmatrix}$ where $\mathbf{A}_i \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{v}_i$ is a vector for all $i \in [1, i^*]$.
  3. Choose random $\mathbf{U} \xleftarrow{R} \mathbb{Z}_q^{n\times L}$.
  4. Sample
     (a) $\mathbf{s}_i \xleftarrow{R} \mathbb{Z}_q^n$ for all $i \in [i^*+1, k]$
     (b) $\mathbf{e}_i \xleftarrow{R} \tilde{D}_{\mathbb{Z},\sigma}^m$ for all $i \in [i^*+1, k]$
     (c) $\mathbf{u}_i \xleftarrow{R} \mathbb{Z}_q^L$ for all $i \in [1, i^*]$
     (d) $d_i \xleftarrow{R} \mathbb{Z}_q$ for all $i \in [k]$
  5. Compute $\mathbf{v}_i^\intercal = \mathbf{s}_i^\intercal \mathbf{A}_i + \mathbf{e}_i^\intercal$ for all $i \in [i^*+1, k]$.
  6. Set $\mathbf{U}'_i = \begin{bmatrix} \mathbf{U} \\ \mathbf{u}_i^\intercal \end{bmatrix}$ for all $i \in [1, i^*]$.
  7. Sample $\mathbf{W}_i \xleftarrow{R} \mathsf{SamplePre}(\mathbf{A}_i, \mathsf{td}_i, \mathbf{U}, \sigma)$ for all $i \in [i^*+1, k]$.
  8. Sample $\mathbf{W}_i \xleftarrow{R} \mathsf{SamplePre}(\mathbf{A}'_i, \mathsf{td}_i, \mathbf{U}'_i, \sigma)$ for all $i \in [1, i^*]$.
  9. Output crs $= \{\mathbf{U}, \mathbf{A}_i, \mathbf{W}_i, \mathbf{v}_i, d_i\}_{i\in[k]}$.

We now state a sequence of claims which state that the advantage of any polytime attacker in one game will be negligibly close to the adjacent game. All but one of our claims are statistical and derive from an immediate application of our lattice preliminaries of 3.4. The remaining claim follows from a straightforward reduction to the learning with errors assumption. We provide the proofs for these in Appendix A.

Claim 6.1. *By the preimage sampling property of Lemma 3.5 the advantage of any algorithm $\mathcal{A}$ in* $\mathsf{Game}_{i^*,2}$ *is negligibly close to its advantage in* $\mathsf{Game}_{i^*,1}$ *for* $i^* \in [k]$.

Claim 6.2. *By the trapdoor distribution property of Theorem 3.4 the advantage of any algorithm $\mathcal{A}$ in* $\mathsf{Game}_{i^*,3}$ *is negligibly close to its advantage in* $\mathsf{Game}_{i^*,2}$ *for* $i^* \in [k]$.

Claim 6.3. *Assuming the* $\mathsf{LWE}_{n,m,q,\sigma}$ *assumption (for n, m, q, σ specified as in the construction) the advantage of any polynomial time algorithm $\mathcal{A}$ in* $\mathsf{Game}_{i^*,4}$ *is negligibly close to its advantage in* $\mathsf{Game}_{i^*,3}$ *for* $i^* \in [k]$.

Claim 6.4. *By the trapdoor distribution property of Theorem 3.4 the advantage of any algorithm $\mathcal{A}$ in* $\mathsf{Game}_{i^*,5}$ *is negligibly close to its advantage in* $\mathsf{Game}_{i^*,4}$ *for* $i^* \in [k]$ *for* $i^* \in [k]$.

Claim 6.5. *By the preimage sampling property of Lemma 3.5 the advantage of any algorithm $\mathcal{A}$ in* $\mathsf{Game}_{i^*,6}$ *is negligibly close to its advantage in* $\mathsf{Game}_{i^*,5}$ *for* $i^* \in [k]$.

Claim 6.6. *The advantage of any algorithm $\mathcal{A}$ in* $\mathsf{Game}_{i^*,7}$ *is identical to its advantage in* $\mathsf{Game}_{i^*,6}$ *for* $i^* \in [k]$.

Claim 6.7. *The advantage of any algorithm $\mathcal{A}$ in* $\mathsf{Game}_{i^*+1,1}$ *is identical to its advantage in* $\mathsf{Game}_{i^*,7}$ *for* $i^* \in [1, k-1]$.

Claim 6.8. *The algorithm* Setup *from our construction has the same distribution as the setup described in* $\mathsf{Game}_{i^*=1,1}$.

Claim 6.9. *The algorithm* SetupHiding *from our construction has the same distribution as the setup described in* $\mathsf{Game}_{i^*=k,7}$.

Lemma 6.10. *Assuming the Assuming the* $\mathsf{LWE}_{n,m,q,\sigma}$ *assumption the advantage of any polynomial time attacker $\mathcal{A}$ against our construction in the single bit hiding game of Definition 2.3 is negligibly close to its advantage when playing the single bit hiding game using* SetupHiding *used in place of* Setup.

Proof. The proof follows from the Claims 6.1 to 6.9. Since k is polynomial in the security parameter and Claims 6.1 to 6.7 show the advantage in successive games is negligibly close, then the advantage of any polynomial time attacker in $\mathsf{Game}_{i^*=1,1}$ is negligibly close to its advantage in $\mathsf{Game}_{i^*=k,7}$. Claim 6.8 shows that the output of $\mathsf{Game}_{i^*=1,1}$ is distributed as the Setup algorithm from the construction. Claim 6.9 shows that the output of $\mathsf{Game}_{i^*=k,7}$ is distributed the same as the SetupHiding algorithm.

Therefore the outputs of Setup and SetupHiding are computationally indistinguishable. It immediately follows that the advantage of any polynomial time attacker in playing the single bit hiding game with a crs derived from the output of Setup is negligibly close to its advantage when playing the single bit hiding game when the crs is derived from SetupHiding.

□

7 Single Bit Hiding

In Section 6 we proved indistinguishability of two setup modes. We move to showing that for any $i^* \in [k]$ the bit r_{i^*} will be statistically hidden. We accomplish this by arguing when crs is sampled from SetupHiding with high probability there will exist a short vector $\mathbf{c}$ with the following properties. The first is that $\mathbf{W}_i\mathbf{c} = \mathbf{0}$ for all $i \neq i^*$ and $\mathbf{U}\mathbf{c} = 0$. Second, $\mathbf{v}_{i^*}^\intercal \mathbf{W}_{i^*}\mathbf{c} = \lfloor q/2 \rfloor$.

Now consider a challenger that flips a random bit $\delta \in \{0, 1\}$ and chooses its randomness as $\mathbf{t} + \delta\mathbf{c}$. Since $\mathbf{c}$ is small and $\mathbf{t}$ is chosen from $[-B, B]$ the distribution is the statistically close to just choosing from $\mathbf{t}$ so the attacker's advantage must be close to the earlier game. The commitment com and proofs π_i for $i \neq i^*$ are all independent of the bit δ as $\mathsf{com} = \mathbf{U}(\mathbf{t} + \delta\mathbf{c}) = \mathbf{U}\mathbf{t} + \delta\mathbf{U}\mathbf{c} = \mathbf{U}\mathbf{t}$ and $\pi_i = \mathbf{W}_i(\mathbf{t} + \delta\mathbf{c}) = \mathbf{W}_i\mathbf{t} + \delta\mathbf{W}_i\mathbf{c} = \mathbf{W}_i\mathbf{t}$. At the same time

$$r_{i^*} = \left\lfloor \mathbf{v}_{i^*}^\intercal \pi_{i^*} + d_i \right\rceil = \left\lfloor \mathbf{v}_{i^*}^\intercal \mathbf{W}_{i^*}(\mathbf{t} + \delta\mathbf{c}) + d_i \right\rceil = \left\lfloor \mathbf{v}_{i^*}^\intercal \mathbf{W}_{i^*}\mathbf{t} + d_i + \delta\lfloor q/2 \rfloor \right\rceil = \left\lfloor \mathbf{v}_{i^*}^\intercal \mathbf{W}_{i^*}\mathbf{t} + d_i \right\rceil \oplus \delta.$$

That is the bit is distributed uniformly at random. We proceed by formally describing both parts of our proof by a new sequence of games. Please note that the game labeling here is independent from that of Section 6. Also note that in the previous section the variable i^* was used to denote an index in a game. In the game sequence below i^* will be the index for bit the attacker is trying to guess in the single bit hiding game.

- Game_1: The single bit hiding game of Definition 2.3 on our construction with the exception that the crs is derived from the SetupHiding algorithm.
- Game_2: Same as Game_1 except after $\mathsf{SetupHiding}(1^\lambda, 1^k) \rightarrow \mathsf{crs} = (\mathbf{A}_i, \mathbf{W}_i, \mathbf{v}_i, d_i\}_{i\in[k]})$ is run the challenger finds a length L vector $\mathbf{c}$. The vector $\mathbf{c}$ will have the following properties: (1) it is short in that $\|\mathbf{c}\|_\infty \leq 5\lambda + 1$, (2) $\mathbf{W}_i\mathbf{c} = \mathbf{0}$ for all $i \neq i^*$ and (3) $\mathbf{u}_{i^*}^\intercal \mathbf{c} = \lfloor q/2 \rfloor$ where $\mathbf{u}_{i^*}$ is a vector sampled in SetupHiding. If no such vector $\mathbf{c}$ exists, the game aborts after assigning the attacker a random guess b'. Otherwise it continues on as in Game_1.
- Game_3: Same as Game_2 except we modify the GenBits algorithm used by challenger in the game as follows.
  1. Sample $\mathbf{t} \xleftarrow{R} [-B, B]^L$.
  2. Sample a random bit $\delta \in \{0, 1\}$.
  3. Compute $\pi_i = \mathbf{W}_i(\mathbf{t} + \delta\mathbf{c})$ for all $i \in [k]$.
  4. Set $r_i = \lfloor \mathbf{v}^\intercal \pi_i + d_i \rceil$ for all $i \in [k]$.
  5. Set $\mathsf{com} = \mathbf{A}_1\mathbf{W}_1(\mathbf{t} + \delta\mathbf{c})$.
  6. Output $(\mathsf{com}, \mathbf{r}, (\pi_1, \ldots, \pi_k))$.

Lemma 7.1. *The advantage of any algorithm $\mathcal{A}$ in Game_2 is negligibly close to its advantage in Game_1.*

Proof. We note that Game_2 proceeds the same as Game_1 except for the fact that it searchers for the vector $\mathbf{c}$ and aborts if no vector meeting these conditions exists. To prove the advantages are negligibly close we must prove that the vector will exist with all but negligible probability. We begin the proof with a supporting claim.

Claim 7.2. *In Game_1 there exists a sequence of vectors $\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_{5\lambda} \in \{-1, 0, 1\}^L$ and unique indices $\mathsf{ind}_0, \mathsf{ind}_1, \ldots, \mathsf{ind}_{5\lambda} \in [k]$ with the following properties:*

- $\mathbf{W}_i\mathbf{h}_j = \mathbf{0}$ *for all* $i \neq i^*$
- *For all* $j \in [0, 5\lambda]$ *we have* $\mathbf{h}_j[\mathsf{ind}_j] = 1$
- *For all* $j, j' \in [0, 5\lambda]$ *where* $j' > j$ *we have* $\mathbf{h}_{j'}[\mathsf{ind}_j] = 0$.
- *The vectors* $\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_{5\lambda}$ *are linearly independent.*

The properties are defined over the matrices $\mathbf{W}_i$ output from setup.

Proof. We prove the claim by defining an algorithm that will produce such vectors and then analyzing it.

$\mathsf{FindVectors}(\{\mathbf{W}_i\}_{i\neq i^*})$

1. Initialize set $S = \emptyset$.
2. For $j = 0$ to 5λ
   (a) Let $\mathbf{h}_j$ be the lexiographically first vector in $\{-1, 0, 1\}^L$ such that: (1) $\mathbf{W}_i\mathbf{h}_j = \mathbf{0}$ for all $i \neq i^*$, (2) at least one entry of $\mathbf{h}_j$ is 1 and (3) for all $z \in S$ we have $\mathbf{h}_j[z] = 0$.
   (b) Set ind_j to be the smallest $z \in [L]$ such that $\mathbf{h}_j = 1$.
   (c) Add the index ind_j to the set S.
3. Output the sequence $\mathbf{h}_0, \ldots, \mathbf{h}_{5\lambda}$ and $\mathsf{ind}_0, \ldots, \mathsf{ind}_{5\lambda}$

Consider iteration j of the algorithm where the algorithm so far has produced vectors $\mathbf{h}_1, \ldots, \mathbf{h}_{j-1}$ and unique indices $\mathsf{ind}_1, \ldots, \mathsf{ind}_{j-1}$ that so far satisfy the conditions above. In addition, assume that $S = \mathsf{ind}_1 \cup \mathsf{ind}_2 \cdots \cup \mathsf{ind}_{j-1}$.

We first show that there are at least two vectors $\mathbf{y}_0, \mathbf{y}_1 \in \{0,1\}$ such that (1) $\mathbf{y}_0 \neq \mathbf{y}_1$, (2) $\mathbf{W}_i\mathbf{y}_0 = \mathbf{W}_i\mathbf{y}_1$ for all $i \neq i^*$ and (3) $\mathbf{y}_0[z] = \mathbf{y}_1[z] = 0$ for all $z \in S$.

We first count the number of possible outputs formed by mutliplying a vector $\mathbf{y}$ by all $\mathbf{W}_i$ for $i \neq i^*$. Consider the process of multiplying a particular matrix $\mathbf{W}_i$ (for $i \neq i^*$) by a vector $y \in \{0,1\}^L$. The absolute value of each entry in the matrix $\mathbf{W}_i$ is at most $\sigma\sqrt{\lambda}$. After multiplication by a bit vector of length L this results in a row vector of m entries where each entry must fall within $[-L\sigma\sqrt{\lambda}, L\sigma\sqrt{\lambda}]$. Since L, σ, λ are polynomial there are at most 2^λ possible values for each entry. [1] If we now consider a vector $\mathbf{y}$ being multiplied by every $\mathbf{W}_i$ for $i \neq i^*$, we can see that there are at most $2^{\lambda(k-1)m} < 2^{\lambda km}$ possibly outputs.

Now we count the number of possible bit vectors that are allowed by our constraints. There are 2^L possible bit vectors of length L. If we restrict ourselves to bit vectors $\mathbf{y}$ where $\mathbf{y}$ where $\mathbf{y}[z] = 0$ for all $z \in S$, then there are $2^{L-j+1} = 2^{\lambda km+5\lambda-j+1}$ of these since $L = \lambda km + 5\lambda + 1$.

We can now see that the number of possible inputs $2^{\lambda km+5\lambda-j+1}$ is greater than the bound on the number of possible outputs $2^{\lambda km}$ since the set S at the beginning of iteration j is of size $j < 5\lambda + 1$ for all possible $j \in [0, 5\lambda]$. By the pigeon hole principle there exists two vectors $\mathbf{y}_0$ and $\mathbf{y}_1$ meeting the above criteria.

We now consider the vector $\mathbf{y}_1 - \mathbf{y}_0 \in \{-1, 0, 1\}$ for such a pair $\mathbf{y}_0$ and $\mathbf{y}_1$. Since $\mathbf{y}_0 \neq \mathbf{y}_1$ the vector $\mathbf{y}_1 - \mathbf{y}_0$ must have at least one non-zero entry. in $\{-1, 1\}$. Second we have that $(\mathbf{y}_1 - \mathbf{y}_0)[z] = 0$ for all $z \in S$ by construction. Finally, since $\mathbf{W}_i\mathbf{y}_0 = \mathbf{W}_i\mathbf{y}_1$ for all $i \neq i^*$ we have $\mathbf{W}_i(\mathbf{y}_1 - \mathbf{y}_0) = \mathbf{0}$. Thus either $(\mathbf{y}_1 - \mathbf{y}_0)$ or $-(\mathbf{y}_1 - \mathbf{y}_0)$ will satisfy the criteria for step 2(a) of the algorithm. [2]

Given that at least one vector exists meeting the criteria exists the Steps 2(a,b) of the (brute force) FindVectors algorithm will successfully find one of them. Since our proof works for all steps $j \in [0, 5\lambda]$ the algorithm will output $5\lambda + 1$ vectors and indices meeting the criteria. The linear independence criteria follows from the second and third properties of the $\mathbf{h}$ vectors.

□

Now that we established vectors $\mathbf{h}_j$ such that $\mathbf{W}_i\mathbf{h}_j = \mathbf{0}$ we want to combine these to construct a vector $\mathbf{c}$ where $\mathbf{u}^\intercal\mathbf{c} = \lfloor q/2 \rfloor$.

Claim 7.3. *Let $\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_{5\lambda} \in \{-1, 0, 1\}^L$ be the vectors output from the* FindVectors *above run on* Game_1. *With all but negligible probability there exists $x_1, \ldots, x_{5\lambda} \in \{0, 1\}$ such that when we let $\mathbf{c} = \mathbf{h}_0 + \Sigma_{j\in[5\lambda]} x_j\mathbf{h}_j$ we have $\mathbf{u}_{i^*}^\intercal\mathbf{c} = \lfloor q/2 \rfloor$.*

Proof. We can view $\mathbf{u}_{i^*}$ and $\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_{5\lambda}$ as defining a hash function $F = F^{\mathbf{u}_{i^*}}_{\mathbf{h}_j : j\in[0,5\lambda]}$. We evaluate the function as

$$F(x_1, \ldots, x_{5\lambda}) = \mathbf{u}_{i^*}^\intercal(\mathbf{h}_0 + \Sigma_{j\in[5\lambda]} x_j\mathbf{h}_j).$$

Since $\mathbf{u}_{i^*}$ is chosen uniformly at random and $\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_{5\lambda}$ are linearly independent and derived independently of $\mathbf{u}_{i^*}$ this forms a pairwise independent hash function family over $\mathbb{Z}_q$. Observe that $\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_{5\lambda}$ are determined by the FindVectors function run of $\{\mathbf{W}_{i^*}\}_{i\neq i^*}$. However, these matrices are constructed independently of $\mathbf{u}_{i^*}$ as the SamplePre operation that constructs them takes in the target $\mathbf{U}$, but that is chosen independently of $\mathbf{u}_{i^*}$.

By the leftover hash lemma [HILL99] a pairwise independent hash function is a strong extractor. The distribution of sampling and outputting a hash function description $\mathbf{u}_{i^*}, \mathbf{h}_0, \ldots, \mathbf{h}_{5\lambda}$ followed by evaluating $F^{\mathbf{u}_{i^*}}_{\mathbf{h}_j : j\in[0,5\lambda]}(\mathbf{x})$ should be statistically close to the distribution of sampling and outputting a hash function description followed by outputting a random element of $\mathbb{Z}_q$. Since $|\mathbb{Z}_q| < 2^{\lambda+1}$ and $\mathbf{x} = x_1, \ldots, x_{5\lambda}$ the statistical difference is $2^{-2\lambda+1}$.

Suppose to the contrary that with non-negligible probability ϵ the hash function $F^{\mathbf{u}_{i^*}}_{\mathbf{h}_j : j\in[0,5\lambda]}(\cdot)$ sampled had the probability that *no* $\mathbf{x}$ exists where $F^{\mathbf{u}_{i^*}}_{\mathbf{h}_j : j\in[0,5\lambda]}(\mathbf{x}) = \lfloor q/2 \rfloor$. Call such a hash function a "bad hash". Consider the event where a description of a bad hash function is sampled followed by $\lfloor q/2 \rfloor$. When the output is sampled by samplding then evaluating the hash this happens with probability 0. When the output is sampled by sampling the hash followed by outputting a random element of $\mathbb{Z}_q$ this happens with probability $\epsilon\frac{1}{q} > \epsilon \cdot 2^{-\lambda+1}$.

[1] We can also achieve a similar bound by noting that operations will be done in $\mathbb{Z}_q$. So even without the bound argument each entry could take on at most $q < 2^{\lambda+1}$ values.

[2] If $(\mathbf{y}_1 - \mathbf{y}_0)$ only has entries in {-1,0}, then $-(\mathbf{y}_1 - \mathbf{y}_0)$ will have at least one 1 entry and meet the other properties.

The statistical difference between the two distributions is then at least $\epsilon \cdot 2^{-\lambda+1}$ which contradicts the leftover hash lemma.

□

The vector $\mathbf{c}$ from the last claim meets the properties of our lemma given out by the last claim along with the fact that $\|\mathbf{c}\|_\infty$ can be at most $5\lambda + 1$ since it is a subset sum of $5\lambda + 1$ vectors in $\{-1, 0, 1\}$.

□

Claim 7.4. *By the smudging lemma 3.2 the advantage of any $\mathcal{A}$ in* Game_3 *is negligibly close to its advantage in* Game_2.

Proof. The only difference in the games is in how we sample the randomness in the GenBits algorithm. In Game_2 the vector $\mathbf{t}$ is chosen from uniformly at random from $[-B, B]$. In Game_3 we replace this by sampling $\mathbf{t}$ from $[-B, B]$ and then adding $\delta\mathbf{c}$ for a randomly chosen $\delta \in \{0, 1\}$. Since $\delta\mathbf{c}$ is a vector with entries in $[-(5\lambda + 1), 5\lambda + 1]$ and B is exponential in λ, the smudging lemma immediately applies. □

Claim 7.5. *The advantage of any $\mathcal{A}$ in* Game_3 *is negligible.*

Proof. The attacker in this game needs to guess the bit $r_{i^*} = \left\lfloor \mathbf{v}_i^\top \pi_{i^*} + d_{i^*} \right\rceil$. We can see.

$$\begin{aligned}
\left\lfloor \mathbf{v}_{i^*}^\top \pi_{i^*} + d_{i^*} \right\rceil &= \left\lfloor \mathbf{v}_{i^*}^\top \mathbf{W}_{i^*}(\mathbf{t} + \delta\mathbf{c}) + d_{i^*} \right\rceil && \text{By definition of the game.} \\
&= \left\lfloor \mathbf{u}_{i^*}^\top (\mathbf{t} + \delta\mathbf{c}) + d_{i^*} \right\rceil && \text{Since } \mathbf{u}_{i^*}^\top = \mathbf{v}_{i^*}^\top \mathbf{W}_{i^*} \text{ in the game.} \\
&= \left\lfloor \mathbf{u}_{i^*}^\top \mathbf{t} + d_{i^*} + \delta\lfloor q/2 \rfloor \right\rceil && \text{Since } \mathbf{u}_{i^*}^\top \mathbf{c} = \lfloor q/2 \rfloor.
\end{aligned}$$

For all but one possible value of $\mathbf{u}_{i^*}^\top \mathbf{t} + d_{i^*}$ in $\mathbb{Z}_q$ we always have that $\left\lfloor \mathbf{u}_{i^*}^\top \mathbf{t} + d_{i^*} + \delta\lfloor q/2 \rfloor \right\rceil = \delta \oplus \left\lfloor \mathbf{u}_{i^*}^\top \mathbf{t} + d_{i^*} \right\rceil$. Therefore except with a negligible probability of $1/q$ the output bit is completely hidden from the adversary *assuming all other information given out to the attacker is independent of δ*.

To establish that no other information on δ is given to the attacker we first see that $\mathsf{com} = \mathbf{U}(\mathbf{t} + \delta\mathbf{c})$. For any $i \neq i^*$ we have that $\mathsf{com} = \mathbf{A}_i \mathbf{W}_i (\mathbf{t} + \delta\mathbf{c})$. But by the setting of $\mathbf{c}$ we have that $\mathbf{W}_i \mathbf{c} = \mathbf{0}$. Thus $\mathsf{com} = \mathbf{A}_i \mathbf{W}_i \mathbf{t}$ and is independent of δ.

Next we check that $\pi_i = \mathbf{W}_i(\mathbf{t} + \delta\mathbf{c})$ for all $i \neq i^*$. Again, $\mathbf{W}_i \mathbf{c} = \mathbf{0}$ and we have that $\pi_i = \mathbf{W}_i \mathbf{t}$ and is also independent of the bit δ. All r_i for $i \neq i^*$ can be determined from the public parameters and π_i so these can give no more information either. Thus everything given to the attacker is independent of δ and the attacker's advantage is negligible. □

Theorem 7.6. *Assuming the Assuming the* $\mathsf{LWE}_{n,m,q,\sigma}$ *assumption our construction is secure in the single-bit hiding game of Definition 2.3.*

Proof. The proof follows immediately from Lemma 6.10, Lemma 7.1, Claim 7.4 and Claim 7.5. □

Acknowledgements

Brent Waters is supported by NSF CNS-1908611, CNS-2318701, and a Simons Investigator award. We thank David Wu for several helpful discussions.

References

[ABB10a] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, pages 553–572, 2010.

[ABB10b] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In *CRYPTO*, pages 98–115, 2010.

[AJL+12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In *EUROCRYPT*, pages 483–501, 2012.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *STOC*, pages 99–108, 1996.

[BCHK07] Dan Boneh, Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. *SIAM J. Comput.*, 36(5):1301–1328, 2007.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO*, pages 213–229. Springer, 2001.

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

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

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

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 97–106. IEEE Computer Society, 2011.

[BWW23] Eli Bradley, Brent Waters, and David J. Wu. Batch arguments to nizks from one-way functions. *IACR Cryptol. ePrint Arch.*, page 1938, 2023.

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

[CCRR18] Ran Canetti, Yilei Chen, Leonid Reyzin, and Ron D. Rothblum. Fiat-shamir and correlation intractability from strong kdm-secure encryption. In *EUROCRYPT*, pages 91–122, 2018.

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

[CGJ+23] Arka Rai Choudhuri, Sanjam Garg, Abhishek Jain, Zhengzhong Jin, and Jiaheng Zhang. Correlation intractability and SNARGs from sub-exponential DDH. In *CRYPTO*, 2023.

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

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT*, pages 523–552, 2010.

[CJJ21a] Arka Rai Choudhuri, Abhishek Jain, and Zhengzhong Jin. Non-interactive batch arguments for NP from standard assumptions. In *CRYPTO*, 2021.

[CJJ21b] Arka Rai Choudhuri, Abhishek Jain, and Zhengzhong Jin. SNARGs for P from LWE. In *FOCS*, 2021.

[CW23] Jeffrey Champion and David J. Wu. Non-interactive zero-knowledge from non-interactive batch arguments. In *CRYPTO*, pages 38–71, 2023.

[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.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 169–178. ACM, 2009.

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

[GMW86] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In *FOCS*, pages 174–187. IEEE Computer Society, 1986.

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In *EUROCRYPT*, 2006.

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

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*, pages 75–92, 2013.

[GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In *STOC*, pages 545–554. ACM, 2013.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[HJKS22] James Hulett, Ruta Jawale, Dakshita Khurana, and Akshayaram Srinivasan. Snargs for P from sub-exponential DDH and QR. In *EUROCRYPT*, 2022.

[HL18] Justin Holmgren and Alex Lombardi. Cryptographic hashing from strong one-way functions (or: One-way product functions and their applications). In *FOCS*, pages 850–858. IEEE Computer Society, 2018.

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

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

[KRR17] Yael Tauman Kalai, Guy N. Rothblum, and Ron D. Rothblum. From obfuscation to the security of fiat-shamir for proofs. In *CRYPTO*, pages 224–251, 2017.

[KW19] Venkata Koppula and Brent Waters. Realizing chosen ciphertext security generically in attribute-based encryption and predicate encryption. In *CRYPTO*, pages 671–700. Springer, 2019.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, pages 700–718, 2012.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC*, pages 427–437. ACM, 1990.

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In *STOC*, pages 333–342, 2009.

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

[PW08] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In *STOC*, 2008.

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

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, pages 84–93, 2005.

[RS91] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *CRYPTO*, 1991.

[WW22] Brent Waters and David J. Wu. Batch arguments for NP and more from standard bilinear group assumptions. In *CRYPTO*, 2022.

A Proofs of Section 6

We now provide the proofs that were omitted earlier from Section 6.

Claim 6.1

Proof. We describe an algorithm $\mathcal{B}$ that plays the pre-image sampling game. Algorithm $\mathcal{B}$ receives from the challenger $\mathbf{A}^*, \mathbf{W}^*, \mathbf{U}^*$. It then proceeds to run $\mathsf{Game}_{i^*,1}$ for attacker $\mathcal{A}$, but with the following exceptions: (1) it sets $\mathbf{U} = \mathbf{U}^*$, (2) it sets $\mathbf{A}_{i^*} = \mathbf{A}^*$ and (3) it sets $\mathbf{W}_{i^*} = \mathbf{W}^*$ instead of computing it with $\mathsf{SamplePre}$.

If the challenger's coin $\beta = 0$, then $(\mathbf{A}^*, \mathsf{td}^*) \leftarrow \mathsf{TrapGen}(1^n, q, m)$, $\mathbf{U}^* \xleftarrow{R} \mathbb{Z}_q^{n \times L}$ and $\mathbf{W}^* \xleftarrow{R} \mathsf{SamplePre}(\mathbf{A}^*, \mathsf{td}^*, \mathbf{U}^*, \sigma)$. By inspection of our assignments we can see that this exactly emulates $\mathsf{Game}_{i^*,1}$.

If the challenger's coin $\beta = 1$, then $(\mathbf{A}^*, \mathsf{td}^*) \leftarrow \mathsf{TrapGen}(1^n, q, m)$, $\mathbf{W}^* \xleftarrow{R} D_{\mathbb{Z},\sigma}^{m \times L}$ and $\mathbf{U}^* = \mathbf{A}^*\mathbf{W}^*$ By inspection of our assignments we can see that this exactly emulates $\mathsf{Game}_{i^*,2}$. If an attacker $\mathcal{A}$ has a non-negligible difference in advantage between $\mathsf{Game}_{i^*,1}$ and $\mathsf{Game}_{i^*,2}$, then $\mathcal{B}$ has a non-negligible advantage in the pre-image sampling game. □

Claim 6.2

Proof. We describe an algorithm $\mathcal{B}$ that plays the trapdoor distribution game. Algorithm $\mathcal{B}$ receives from the challenger $\mathbf{A}^*$. It then proceeds to run $\mathsf{Game}_{i^*,2}$ for attacker $\mathcal{A}$, but with the following exception: it sets $\mathbf{A}_{i^*} = \mathbf{A}^*$. We remark that the trapdoor for $\mathbf{A}_{i^*}$ is not used in this game, so $\mathcal{B}$ can proceed without it.

If the challenger's coin $\beta = 0$, then $(\mathbf{A}^*, \mathsf{td}^*) \leftarrow \mathsf{TrapGen}(1^n, q, m)$. By inspection of our assignments we can see that this exactly emulates $\mathsf{Game}_{i^*,2}$.

If the challenger's coin $\beta = 1$, then $\mathbf{A}^* \leftarrow \mathbb{Z}_q^{n \times m}$. By inspection of our assignments we can see that this exactly emulates $\mathsf{Game}_{i^*,3}$. If an attacker $\mathcal{A}$ has a non-negligible difference in advantage between $\mathsf{Game}_{i^*,2}$ and $\mathsf{Game}_{i^*,3}$, then $\mathcal{B}$ has a non-negligible advantage in the trapdoor distribution game.

□

Claim 6.3

Proof. We describe an algorithm $\mathcal{B}$ that plays the decisional $\mathsf{LWE}_{n,m,q,\sigma}$ game. Algorithm $\mathcal{B}$ receives from the challenger $\mathbf{A}^*, \mathbf{v}^*$. It then proceeds to run $\mathsf{Game}_{i^*,3}$ for attacker $\mathcal{A}$, but with the following exceptions: (1) it sets $\mathbf{A}_{i^*} = \mathbf{A}^*$ and (2) it sets $\mathbf{v}_{i^*} = \mathbf{v}^*$.

If the challenger's coin $\beta = 0$, then $\mathbf{A}^* \leftarrow \mathbb{Z}_q^{n \times m}$, $\mathbf{s}^* \xleftarrow{R} \mathbb{Z}_q^n$, $\mathbf{e}^* \xleftarrow{R} \tilde{D}_{\mathbb{Z},\sigma}^m$ and $\mathbf{v}^* = (\mathbf{s}^*)^\intercal \mathbf{A}^* + (\mathbf{e}^*)^\intercal$ By inspection of our assignments we can see that this exactly emulates $\mathsf{Game}_{i^*,3}$.

If the challenger's coin $\beta = 1$, then $\mathbf{A}^* \leftarrow \mathbb{Z}_q^{n \times m}$ and $\mathbf{v}^* \xleftarrow{R} \mathbb{Z}_q^m$. By inspection of our assignments we can see that this exactly emulates $\mathsf{Game}_{i^*,4}$. Note that the additional notation of defining $\mathbf{u}_{i^*}$ does not impact the game since $\mathbf{u}_{i^*}$ is not used anywhere else in the algorithm.

If an attacker $\mathcal{A}$ has a non-negligible difference in advantage between $\mathsf{Game}_{i^*,3}$ and $\mathsf{Game}_{i^*,4}$, then $\mathcal{B}$ has a non-negligible advantage in the trapdoor distribution game.

□

Claim 6.4

Proof. We describe an algorithm $\mathcal{B}$ that plays the trapdoor distribution game. Algorithm $\mathcal{B}$ receives from the challenger $\mathbf{A}'^* \in \mathbb{Z}_q^{n+1\times m}$. It then proceeds to run $\mathsf{Game}_{i^*,4}$ for attacker $\mathcal{A}$, but with the following exception: parse and assign $\mathbf{A}'^* = \begin{bmatrix} \mathbf{A}_{i^*} \\ \mathbf{v}_{i^*}^\intercal \end{bmatrix}$. We again remark that the trapdoor for $\mathbf{A}_{i^*}$ is not used in this game, so $\mathcal{B}$ can proceed without it.

If the challenger's coin $\beta = 1$, then $\mathbf{A}'^* \leftarrow \mathbb{Z}_q^{n+1\times m}$ and both $\mathbf{A}_{i^*}$ and $\mathbf{v}_{i^*}$ are chosen uniformly at random matching the distribution of $\mathsf{Game}_{i^*,4}$.

If the challenger's coin $\beta = 0$, then $(\mathbf{A}'^*, \mathsf{td}^*) \leftarrow \mathsf{TrapGen}(1^{n+1}, q, m)$. By inspection of our assignments we can see that this exactly emulates $\mathsf{Game}_{i^*,5}$.

If an attacker $\mathcal{A}$ has a non-negligible difference in advantage between $\mathsf{Game}_{i^*,4}$ and $\mathsf{Game}_{i^*,5}$, then $\mathcal{B}$ has a non-negligible advantage in the trapdoor distribution game.

□

Claim 6.5

Proof. We describe an algorithm $\mathcal{B}$ that plays the pre-image sampling game. Algorithm $\mathcal{B}$ receives from the challenger $\mathbf{A}'^* \in \mathbb{Z}_q^{n+1\times m}, \mathbf{W}^*, \mathbf{U}'^* \in \mathbb{Z}_q^{n+1\times m}$. It then proceeds to run $\mathsf{Game}_{i^*,5}$ for attacker $\mathcal{A}$, but with the following exceptions: (1) it sets $\begin{bmatrix} \mathbf{U} \\ \mathbf{u}_{i^*}^\intercal \end{bmatrix} = \mathbf{U}'_{i^*} = \mathbf{U}'^*$, (2) it sets $\mathbf{A}'_{i^*} = \mathbf{A}'^*$ and (3) it sets $\mathbf{W}_{i^*} = \mathbf{W}^*$.

If the challenger's coin $\beta = 1$, then $(\mathbf{A}'^*, \mathsf{td}^*) \leftarrow \mathsf{TrapGen}(1^{n+1}, q, m)$, $\mathbf{W}^* \xleftarrow{\mathrm{R}} D_{\mathbb{Z},\sigma}^{m\times L}$ and $\mathbf{U}'^* = \mathbf{A}'^*\mathbf{W}^*$. By inspection of our assignments we can see that this exactly emulates $\mathsf{Game}_{i^*,5}$.

If the challenger's coin $\beta = 0$, then $(\mathbf{A}'^*, \mathsf{td}^*) \leftarrow \mathsf{TrapGen}(1^{n+1}, q, m), \mathbf{U}'^* \xleftarrow{\mathrm{R}} \mathbb{Z}_q^{n+1\times L}$ and $\mathbf{W}^* \xleftarrow{\mathrm{R}} \mathsf{SamplePre}(\mathbf{A}'^*, \mathsf{td}^*, \mathbf{U}'^*, \sigma)$. By inspection of our assignments we can see that this exactly emulates $\mathsf{Game}_{i^*,6}$.

If an attacker $\mathcal{A}$ has a non-negligible difference in advantage between $\mathsf{Game}_{i^*,5}$ and $\mathsf{Game}_{i^*,6}$, then $\mathcal{B}$ has a non-negligible advantage in the pre-image sampling game.

□

Claim 6.6

Proof. The distribution of these games is identical. The first difference is that $\mathbf{W}_{i^*} \xleftarrow{\mathrm{R}} \mathsf{SamplePre}(\mathbf{A}'_{i^*}, \mathsf{td}_{i^*}, \mathbf{U}'_{i^*}, \sigma)$ is moved to further on down in the setup when it is incorporated into a line that captures sampling all $i \in [1, i^*]$ in this way. However, deferring the sampling does not impact the distribution as $\mathbf{W}_{i^*}$ is not used for sampling other parameters. The second difference is that instead of sampling $\begin{bmatrix} \mathbf{U} \\ \mathbf{u}_{i^*}^\intercal \end{bmatrix}$ in one go as a matrix from $\mathbb{Z}_q^{n+1\times L}$ it samples the matrix $\mathbf{U}$ first and the vector $\mathbf{u}_{i^*}$ later on. However, these result in the same distribution. And $\mathbf{u}_{i^*}$ is defined by the time it is needed in SamplePre.

□

Claim 6.8

Proof. Since $i^* = 1$ any conditions of the form " for all $i \in [i^*, k]$" are equivalent to "for all $i \in [k]$". Moreover, any conditions of the form "for all $i \in [1, i^* - 1]$" refer to an empty set and are never activated. We can observe once these substitutions are made to $\mathsf{Game}_{i^*=1,1}$ it results in the Setup algorithm from our construction. In particular, from $\mathsf{Game}_{i^*=1,1}$ we delete Steps 2, 4(c), 6 and 8 . And from $\mathsf{Game}_{i^*=1,1}$ Steps 1, 4(a,b), 5,7 are changed to "for all $i \in k$". □

Claim 6.9

Proof. Since $i^* = k$ any conditions of the form " for all $i \in [1, i^*]$" are equivalent to "for all $i \in [k]$". Moreover, any conditions of the form "for all $i \in [i^* + 1, k]$" refer to an empty set and are never activated. We can observe once these substitutions are made to $\text{Game}_{i^*=k,7}$ it results in the SetupHiding algorithm from this section. In particular, from $\text{Game}_{i^*=k,7}$ we delete Steps 1, 4(a,b), 5,7. And from $\text{Game}_{i^*=k,7}$ Steps 2, 4(c), 6,8 are changed to "for all $i \in k$". □

B An Alternative Proof of Flexible Binding

In Section 5 we saw a simple proof that the size of the possible revealed strings $\mathcal{V}^{\text{crs}}$ is bounded by the number of possible commitments. This is due to the fact that every CRS and commitment string com *perfectly* binds to (at most) a single string in $\{0,1\}^k$. In turn this is due to the fact that our Verify algorithm in Section Section 4 rejects any proof that lands too close to a rounding boundary. (See Step 4.)

In this section we show an alternative proof of binding, which will work on our construction even if the boundary closeness test of Step 4 is omitted. Intuitively, without such a boundary check there can exist a commitment com for which there are multiple indices $i \in k$ for which an attacker could open to either a zero or one. However, we can show that with all but negligible probability over the choice of crs, there will not exist a commitment com for which an attacker can equivocate on "too many" of the bits. In particular, we will bound the number of flippable locations for a given com by $N = n(\lambda + 1) + \lambda$. Taking into account that there are at most $n(\lambda + 1)$ com values this limits $\mathcal{V}^{\text{crs}} \leq 2^{(n(\lambda+1))^2+\lambda}$ as needed.

As a historical note when writing this work we first devised the argument below and later realized that if we added a boundary check we could use the simpler argument presented in Section 5. We mostly include this proof in the appendix for intellectual curiosity. However, we remark that variations of this analysis might be useful for removing the need of a subexponential modulus to noise ratio for soundness. Although it is less clear how to remove it from the analysis of hiding security.

Definition B.1. Consider the experiment of running the $\text{Setup}(1^\lambda, 1^k)$ algorithm of our Π_{HBG} construction that produces $\text{crs} = \{\mathbf{U}, \mathbf{A}_i, \mathbf{W}_i, \mathbf{v}_i, d_i\}_{i\in[k]}$. Fix a given λ, k and commitment value com. We define the event $\text{FlipRange}_{\text{com},i}$ to be when the crs produced has the property that there exists a values $f_0, f_1 \in [-\texttt{TestBound}\sqrt{\lambda}\sigma m, \texttt{TestBound}\sqrt{\lambda}\sigma m]$ such that $\lfloor \mathbf{s}_i^\intercal \text{com} + f_0 + d_i \rceil = 0$ and $\lfloor \mathbf{s}_i^\intercal \text{com} + f_1 + d_i \rceil = 1$.

We define the event $\text{ExceedRange}^N_{\text{com}}$ to be the event where $\text{FlipRange}_{\text{com},i}$ occurs for at more than N distinct values of $i \in [k]$.

Claim B.2. *The probability of event* $\text{FlipRange}_{\text{com},i}$ *is* $(4 \cdot \texttt{TestBound} \cdot \sqrt{\lambda}\sigma m)/2^\lambda$ *for all* $i \in [k]$. *Moreover for any* $i \neq i'$ *the events* $\text{FlipRange}_{\text{com},i}$ *and* $\text{FlipRange}_{\text{com},i'}$ *are independent.*

Proof. For the round function there exists a single value $a_0 \in \mathbb{Z}_q$ such that $\lfloor a_0 \rceil = 0$, but $\lfloor a_0 + 1 \rceil = 1$. And a different value a_1 such that $\lfloor a_1 \rceil = 1$, but $\lfloor a_1 + 1 \rceil = 0$. For the event $\text{FlipRange}_{\text{com},i}$ to occur it must be the case that either $a_0 \in [d_i + \mathbf{s}_i^\intercal \cdot \text{com} - \texttt{TestBound}\sqrt{\lambda}\sigma m,\ d_i + \mathbf{s}_i^\intercal \cdot \text{com} + \texttt{TestBound}\sqrt{\lambda}\sigma m - 1]$ or $a_1 \in [d_i + \mathbf{s}_i^\intercal \cdot \text{com} - \texttt{TestBound}\sqrt{\lambda}\sigma m,\ d_i + \mathbf{s}_i^\intercal \cdot \text{com} + \texttt{TestBound}\sqrt{\lambda}\sigma m - 1]$. Since d_i is chosen uniformly at random in $\mathbb{Z}_q$ the probability of the first condition is $(2 \cdot \texttt{TestBound} \cdot \sqrt{\lambda}\sigma m)/q$ as is the second. Moreover by the distance between a_0, a_1 these two conditions are mutually exclusive and the probability that either the first or second occurs is therefore $(4 \cdot \texttt{TestBound} \cdot \sqrt{\lambda}\sigma m)/q$ if we assume $2 \cdot \texttt{TestBound} \cdot \sqrt{\lambda}\sigma m < q/2$. Since $q > 2^\lambda$ the probability condition holds.

Finally, the events $\text{FlipRange}_{\text{com},i}$ and $\text{FlipRange}_{\text{com},i'}$ are independent for $i \neq i'$. Suppose we fix all outputs of setup except $d_{i'}$ to some specified values. Note this fixes whether the event $\text{FlipRange}_{\text{com},i}$ occurred or not. The probability that either a_0 or a_1 is in $[d_i + \mathbf{s}_i^\intercal \cdot \text{com} - \texttt{TestBound}\sqrt{\lambda}\sigma m,\ d_i + \mathbf{s}_i^\intercal \cdot \text{com} + \texttt{TestBound}\sqrt{\lambda}\sigma m - 1]$ is still $(4\texttt{TestBound}\sqrt{\lambda}\sigma m)/q$ since there are $4\texttt{TestBound}\sqrt{\lambda}$ values of $d_{i'}$ that satisfy the condition.

□

Claim B.3. *Consider a binding security game against an attacker* $\mathcal{A}$ *which provides the parameter* k *which is bounded by some polynomial in* λ. *Then for all* com *(in legal range) the event* $\text{ExceedRange}^{N=n(\lambda+1)+\lambda}_{\text{com}}$ *happens with probability at most* $2^{-N} = 2^{-(n(\lambda+1)+\lambda)}$ *for all* $\lambda > \lambda_0$ *for some* λ_0.

Proof. Since the events $\mathsf{FlipRange}_{\mathsf{com},i}$ are independent for all i we can assign X_i as the random variable that is 1 when $\mathsf{FlipRange}_{\mathsf{com},i}$ happens and 0 otherise. Then $X = \Sigma_{i=1}^{k} X_i$ is a random variable representing their sum and is distributed according to a binomial distribution with k trials and probability $p = (4 \cdot \mathtt{TestBound} \cdot \sqrt{\lambda}\sigma m)/q$. The mean of the distribution is $\mu = (4k \cdot \mathtt{TestBound} \cdot \sqrt{\lambda}\sigma m)/q$.

From Chernoff bounds we have that for $\delta > 2e - 1$

$$\Pr[X > (1+\delta)\mu] < 2^{-(1+\delta)\mu}.$$

To bound the probability we let $(1+\delta)\mu = N = n(\lambda+1)+\lambda$ which means there will be at most 2^{-N} probability that $X > N$. To finish the proof we need to verify that $\delta > 2e - 1$. Recall that $\mu = (4k \cdot \mathtt{TestBound} \cdot \sqrt{\lambda}\sigma m)/q$. For k which is polynomial in λ the expression $4k \cdot \mathtt{TestBound} \cdot \sqrt{\lambda}\sigma m$ consists of $2^{.5\lambda}$ multiplied by terms which are polynomial in λ. Thus when we divide it by $q > 2^{\lambda}$ we get that $\mu < 1$ for all λ greater than some λ_0. And it is easy to see the condition $\delta > 2e - 1$ holds.

□

We now move to connecting the range events defined above to events where the attacker can decommit to either bit. At a high level we will show that if the $\mathsf{FlipRange}_{\mathsf{com},i}$ event does *not* occur the attacker will be able to open up r_i to at most one value.

Definition B.4. Consider running the $\mathsf{Setup}(1^{\lambda}, 1^{k})$ algorithm of our Π_{HBG} construction that produces $\mathsf{crs} = \{\mathbf{U}, \mathbf{A}_i, \mathbf{W}_i, \mathbf{v}_i, d_i\}_{i\in[k]}$. We define the event $\mathsf{Flippable}_{\mathsf{com},i}$ to occur if there exists π^0, π^1 such that $\mathsf{Verify}(\mathsf{crs}, \mathsf{com}, i, 0, \pi^0) \to 1$ and $\mathsf{Verify}(\mathsf{crs}, \mathsf{com}, i, 1, \pi^1) \to 1$. And the two proofs produce different results for r_i. That is $\lfloor \mathbf{v}_i^{\intercal}\pi^0 + d_i \rceil \neq \lfloor \mathbf{v}_i^{\intercal}\pi^1 + d_i \rceil$.

The event $\mathsf{Flippable}^{N}_{\mathsf{com}}$ is the event where $\mathsf{Flippable}_{\mathsf{com},i}$ occurs for at more than N distinct values of i.

Claim B.5. *If the event* $\mathsf{FlipRange}_{\mathsf{com},i}$ *does not occur in an experiment, then the event* $\mathsf{Flippable}_{\mathsf{com},i}$ *does not occur either.*

Proof. Consider a proof π such that $\mathsf{Verify}(\mathsf{crs}, \mathsf{com}, i, \beta, \pi) \to 1$ for some β. Since the proof verifies we have that $\mathsf{com} = \mathbf{A}_i\pi$. It follows that

$$\lfloor \mathbf{v}_i^{\intercal}\pi + d_i \rceil = \lfloor (\mathbf{s}_i^{\intercal}\mathbf{A}_i + \mathbf{e}_i^{\intercal})\pi + d_i \rceil = \lfloor \mathbf{s}_i^{\intercal}\mathsf{com} + \mathbf{e}_i^{\intercal}\pi + d_i \rceil.$$

Next we bound the value that $\mathbf{e}_i^{\intercal}\pi$ can take. Each entry of $\mathbf{e}_i$ produced from setup is at most $\sqrt{\lambda}\sigma$ and since verification passed each entry of π is at most $\mathtt{TestBound}$. Since the vectors are of length m the most it can be is $\mathtt{TestBound} \cdot \sqrt{\lambda}\sigma m$. Likewise the least value of $\mathbf{e}_i^{\intercal}\pi$ is $-\mathtt{TestBound} \cdot \sqrt{\lambda}\sigma m$.

It follows that $\mathbf{v}_i^{\intercal}\pi + d_i = \mathbf{s}_i^{\intercal}\mathsf{com} + f + d_i$ for some $f \in [-\mathtt{TestBound}\sqrt{\lambda}\sigma m, \mathtt{TestBound}\sqrt{\lambda}\sigma m]$. If the event $\mathsf{FlipRange}_{\mathsf{com},i}$ does not occur, then $\lfloor \mathbf{v}_i^{\intercal}\pi + d_i = \mathbf{s}_i^{\intercal}\mathsf{com} + f + d_i \rceil$ takes on the same value for all $f \in [-\mathtt{TestBound}\sqrt{\lambda}\sigma m, \mathtt{TestBound}\sqrt{\lambda}\sigma m]$. And the event $\mathsf{Flippable}_{\mathsf{com},i}$ does not occur. □

Claim B.6. *For all* N *if the event* $\mathsf{ExceedRange}^{N}_{\mathsf{com}}$ *does not occur in an experiment, then the event* $\mathsf{Flippable}^{N}_{\mathsf{com}}$ *does not occur either.*

Proof. By Claim B.5 for all i the event $\mathsf{Flippable}_{\mathsf{com},i}$ occurs only if $\mathsf{FlipRange}_{\mathsf{com},i}$ does. If event $\mathsf{ExceedRange}^{N}_{\mathsf{com}}$ does not occur, then $\mathsf{FlipRange}_{\mathsf{com},i}$ occurred for N or less distinct $i \in [k]$ values. It must also be the case then that $\mathsf{Flippable}_{\mathsf{com},i}$ occurred for N or less $i \in [k]$ values and $\mathsf{Flippable}^{N}_{\mathsf{com}}$ did not occur either. □

Corollary B.7. *Consider a binding security game against an attacker* $\mathcal{A}$ *which provides a* k *parameter which is bounded by some polynomial in* λ*. For all* com *the event* $\mathsf{Flippable}^{N=n(\lambda+1)+\lambda}_{\mathsf{com}}$ *happens with probability at most* $2^{-N} = 2^{-n(\lambda+1)-\lambda}$ *for all* $\lambda > \lambda_0$ *for some* λ_0*.*

Proof. Follows immediately from the combination of Claims B.3 and B.6. □

Theorem B.8. *With all but negligible probability over the choice of* crs *there will not exist a commitment* com *where the event* $\mathsf{Flippable}^{N=n(\lambda+1)+\lambda}_{\mathsf{com}}$ *occurs.*

Proof. By Claim B.7 $\mathsf{Flippable}_{\mathsf{com}}^{N=n(\lambda+1)+\lambda}$ happens for a particular commitment string com with probability at most $2^{-N} = 2^{-n(\lambda+1)-\lambda}$ for all $\lambda > \lambda_0$. Any commitment string com is an n-length vector of $\mathbb{Z}_q$ where $q < 2^{\lambda+1}$. Thus there are at most $2^{n(\lambda+1)}$ possible com values in the experiment.

It follows from the union bound that the probability there is *any* com where $\mathsf{ExceedRange}_{\mathsf{com}}^{N=n(\lambda+1)+\lambda}$ occurs is bounded by $2^{-\lambda}$ for all $\lambda > \lambda_0$. □