Separating Semantic and Circular Security for Symmetric-Key Bit Encryption from the Learning with Errors Assumption

Rishab Goyal
rgoyal@cs.utexas.edu

Venkata Koppula
kvenkata@cs.utexas.edu

Brent Waters
bwaters@cs.utexas.edu*

Abstract

In this work we separate private-key semantic security from 1-circular security for bit encryption using the Learning with Error assumption. Prior works used the less standard assumptions of multilinear maps or indistinguishability obfuscation. To achieve our results we develop new techniques for obliviously evaluating branching programs.

1 Introduction

Over the past several years the cryptographic community has given considerable attention to the notion of key-dependent message security. In key dependent security we consider an attacker that gains access to ciphertexts that encrypt certain functions of the secret key(s) of the user(s). Ideally, a system should remain semantically secure even in the presence of this additional information.

One of the most prominent problems in key dependent message security is the case of circular security. A circular secure system considers security in the presence of key cycles. A key cycle of k users consists of k encryptions where the i-th ciphertext ct_i is an encryption of the $i+1$ user's secret key under user i's public key. That is $\mathsf{ct}_1 = \mathsf{Encrypt}(\mathrm{PK}_1, \mathrm{SK}_2), \mathsf{ct}_2 = \mathsf{Encrypt}(\mathrm{PK}_2, \mathrm{SK}_3) \ldots, \mathsf{ct}_k = \mathsf{Encrypt}(\mathrm{PK}_k, \mathrm{SK}_1)$. If a system is k circular secure, then such a cycle should be indistinguishable from an encryption of k arbitrary messages. The notion also applies to secret key encryption systems.

One reason that circular security has received significant attention is that the problem has arisen in multiple applications [16, 27, 2], the most notable is that Gentry [22] showed how a circular secure leveled homomorphic encryption can be bootstrapped to homomorphic encryption that works for circuits of unbounded depth. Stemming from this motivation there have been several positive results[11, 6, 13, 8, 14, 5, 4, 29] that have achieved circular and more general notations of key dependent messages security from a variety of cryptographic assumptions.

On the flip side several works have sought to discover if there exist separations between IND-CPA security and different forms of circular security. That is they sought to develop a system that was *not* circular secure, but remained IND-CPA secure. For the case of 1-circular security achieving such a separation is trivial. The (secret key) encryption system simply tests if the message to be encrypted is equal to the secret key SK, if so it gives the message in the clear; otherwise it encrypts as normal. (This example can be easily extended to public key encryption.) Clearly, such a system is not circular secure and it is easy to show it maintains IND-CPA security. More work is required, however, to achieve separations of length greater than one. Separations were first shown for the case of $k = 2$ length cycles using groups with bilinear maps [1, 17] and later [10] under the Learning with Errors assumption [34]. Subsequently, there existed works that achieved separations for arbitrary length cycles[25, 28], however, these required the use obfuscation. All current candidates of general obfuscation schemes rely on the relatively new primitive of multilinear maps, where many such multilinear map candidates have suffered from cryptanalysis attacks [18, 19]. Most recently and Alamati and Peikert [3]

*Supported by NSF CNS-1228599 and CNS-1414082, DARPA SafeWare, Microsoft Faculty Fellowship, and Packard Foundation Fellowship.

and Koppula and Waters [26] showed separations of arbitrary length cycles from the much more standard Learning with Errors assumption.

Another challenging direction in achieving separations for circular security is to consider encryptions systems where the message consist of a single bit. Separating from IND-CPA is difficult even in the case of cycles of length 1 (i.e. someone encrypts their own secret key). Consider a bit encryption system with keys of length $\ell = \ell(\lambda)$. Suppose an attacker receives an encryption of the secret key in the form of ℓ successive bit by bit encryptions. Can this be detected?

We observe that encrypting bit by bit seems to make detection harder. Our trivial counterexample from above no longer applies since the single bit message cannot be compared to the much longer key. The first work to consider such a separation was due to Rothblum [35] who showed that a separation could be achieved from multilinear maps under certain assumptions. One important caveat, however, to his result was that the level of multilinearlity must be greater than $\log(q)$ where q is the group order. This restriction appears to be at odds with current multilinear map/encoding candidates which are based off of "noisy cryptography" and naturally require a bigger modulus whose log is greater than the number of multiplications allowed. Later, Koppula, Ramchen and Waters [25] showed how to achieve a separation from bit encryption using indistinguishability obfuscation. Again, such a tool is not known from standard assumptions.

In this work we aim to separate semantic security from 1-circular security for bit encryption systems under the Learning with Errors assumption. Our motivation to study this problem is two fold. First, achieving such a separation under a standard assumption will significantly increase our confidence compared to obfuscation or multilinear map-based results. Second, studying such a problem presents the opportunity for developing new techniques in the general area of computing on encrypted data and may lead to other results down the line.

To begin with, we wish to highlight some challenges presented by bit encryption systems that were not addressed in prior work. First, the recent results of [3, 26] both use a form of telescoping cancellation where the encryption algorithm takes in a message and uses this as a 'lattice trapdoor'[24, 30]; if the message contained the needed secret key then it cancels out the public key of an "adjacent" ciphertext. We observe that such techniques require an encryption algorithm that receives the entire secret key at once, and there is no clear path to leverage this in the case where an encryption algorithm receives just a single bit message. Second, while the level restriction in Rothblum's result [35] appeared in the context of multilinear maps, the fundamental issue will transcend to our Learning with Errors solution. Looking ahead we will need to perform a computation where the number of multiplication steps is restricted to be less than $\log(q)$, where here q is the modulus we work in.

1.1 Separations from Learning with Errors

We will now describe our bit encryption scheme that is semantically secure but not circular secure. Like previous works [10, 3, 26], we will take decryption out of the picture, and focus on building an IND-CPA secure encryption scheme where one can distinguish between an encryption of the secret key and encryptions of zeroes.

The two primary ingredients of our construction are low-depth pseudorandom functions (PRFs) and lattice trapdoors. In particular, we require a PRF which can be represented using a permutation branching program of polynomial length and polynomial width.[1] Banerjee, Peikert and Rosen [7] showed how to construct LWE based PRFs that can be represented using $\mathbf{NC}^1$ circuits, and using Barrington's theorem [9], we get PRFs that can be represented using branching programs of polynomial length and width 5.

Next, let us recall the notion of lattice trapdoors. A lattice trapdoor generation algorithm outputs a matrix $\mathbf{A}$ together with a trapdoor $T_{\mathbf{A}}$. The matrix looks uniformly random, while the trapdoor can be

[1] Recall, a permutation branching program of length L and width w has w states at each level, an accepting and rejecting state at the top level. Each level $j \leq L$ has two permutations $\sigma_{j,0}$ and $\sigma_{j,1}$ associated, and there is an input-selector function which determines the input read at each level. The program execution starts at state 1 of level 0. Suppose, at level j, the state is $\mathsf{st} \in [w]$. Let b be the input read at level j. Then, the state at level $j+1$ is $\sigma_{j,b}(\mathsf{st})$. Proceeding this way, the program terminates at level L in either the accepting state or rejection state.

used to compute, for any matrix $\mathbf{U}$, a low norm matrix $\mathbf{S} = \mathbf{A}^{-1}(\mathbf{U})$ such that $\mathbf{A} \cdot \mathbf{S} = \mathbf{U}$.[2] As a result, the matrix $\mathbf{S}$ can be used to 'transform' the matrix $\mathbf{A}$ to another matrix $\mathbf{U}$. In this work, we will be interested in *oblivious sequence transformation* : we want a sequence of matrices $\mathbf{B}_1, \ldots, \mathbf{B}_w$ such that for any sequence of matrices $\mathbf{U}_1, \ldots, \mathbf{U}_w$, we can compute a low norm matrix $\mathbf{S}$ such that $\mathbf{B}_i \cdot \mathbf{S} = \mathbf{U}_i$. Note that the same matrix $\mathbf{S}$ should be able to transform any $\mathbf{B}_i$ to $\mathbf{U}_i$; that is, $\mathbf{S}$ is oblivious of i. This obliviousness property will be important for our solution, and together with the telescoping products/cascading cancellations idea of [26, 3, 23], we get our counterexample.

Oblivious Sequence Transformation We first observe that one can easily obtain oblivious sequence transformation, given standard lattice trapdoors. Consider the following matrix $\mathbf{B}$:

$$\mathbf{B} = \begin{bmatrix} \mathbf{B}_1 \\ \vdots \\ \mathbf{B}_w \end{bmatrix}.$$

Let T denote the trapdoor of $\mathbf{B}$ (we will refer to T as the 'joint trapdoor' of $\mathbf{B}_1$, ..., $\mathbf{B}_w$). Now, given any sequence $\mathbf{U}_1, \ldots, \mathbf{U}_w$, we similarly define a new matrix $\mathbf{U}$ which has the $\mathbf{U}_i$ stacked together, and set $\mathbf{S} = \mathbf{B}^{-1}(\mathbf{U})$. Clearly, this satisfies our oblivious sequence transformation requirement.

Our Encryption Scheme As mentioned before, we will only focus on the setup, encryption and testing algorithms. Let PRF be a pseudorandom function family with keys and inputs of length λ, and output being a single bit. For any input i, we require that the function $\mathrm{PRF}(\cdot, i)$ can be represented using a branching program of length L and width 5 (we choose 5 for simplicity here; our formal description works for any polynomial width w). The setup algorithm chooses a PRF key s. Let nbp be a parameter which represents the number of points at which the PRF is evaluated, and let $t_i = \mathrm{PRF}(s, i)$ for $i \leq \mathsf{nbp}$. Finally, for each $i \leq \mathsf{nbp}$, let $\mathsf{BP}^{(i)}$ denote the branching program that evaluates $\mathrm{PRF}(\cdot, i)$. Each branching program $\mathsf{BP}^{(i)}$ has L levels and 5 possible states at each level. At the last level, there are only two valid states — $\mathsf{acc}^{(i)}$ and $\mathsf{rej}^{(i)}$, i.e. the accepting and rejecting state. For each branching program $\mathsf{BP}^{(i)}$ and level j, there are two state transition functions $\sigma^{(i)}_{j,0}, \sigma^{(i)}_{j,1}$ that decide the transition between states depending upon the input bit read. The setup algorithm also chooses, for each branching program $\mathsf{BP}^{(i)}$, level $j \leq L$ and state $k \leq 5$, a matrix $\mathbf{B}^{(i)}_{j,k}$. At all levels $j \neq L$, the matrices $\mathbf{B}^{(i)}_{j,1}, \ldots, \mathbf{B}^{(i)}_{j,5}$ have a joint trapdoor. At the top level, the matrices satisfy the following relation:

$$\sum_{i \,:\, t_i = 0} \mathbf{B}^{(i)}_{L,\mathsf{rej}^{(i)}} + \sum_{i \,:\, t_i = 1} \mathbf{B}^{(i)}_{L,\mathsf{acc}^{(i)}} = \mathbf{0}.$$

The secret key consists of the PRF key s and $\mathsf{nbp} \cdot L$ trapdoors $T^{(i)}_j$.

The encryption algorithm is designed specifically to distinguish key encryptions from encryptions of zeros. Each ciphertext consists of L sub-ciphertexts, one for each level, and each sub-ciphertext consists of nbp sub-sub-ciphertexts. The sub-sub-ciphertext corresponding to $\mathsf{BP}^{(i)}$ at level j can be used to transform $\mathbf{B}^{(i)}_{j,k}$ to $\mathbf{B}^{(i)}_{j+1,\sigma^{(i)}_{j,0}(k)}$ or $\mathbf{B}^{(i)}_{j+1,\sigma^{(i)}_{j,1}(k)}$, depending on the bit encrypted. This is achieved via oblivious sequence transformation. Let b denote the bit encrypted, and let $\mathbf{D}$ be the matrix constructed by stacking $\{\mathbf{B}^{(i)}_{j,1}, \ldots, \mathbf{B}^{(i)}_{j,5}\}$ according to the permutation $\sigma^{(i)}_{j,b}$. The sub-sub-ciphertext $\mathsf{ct}^{(i)}_j$ for program $\mathsf{BP}^{(i)}$ at level j is simply (a noisy approximation of) ${\mathbf{B}^{(i)}_j}^{-1}(\mathbf{D})$. The ciphertext also includes the base matrices $\{\mathbf{B}^{(i)}_0\}$ for each program.

The testing algorithm is used to distinguish between an encryption of the secret key and encryptions of zeros. It uses the first $|s| = \lambda$ ciphertexts, which are either encryptions of the PRF key s, or encryptions of

[2]For simplicity, we use the notation $\mathbf{A}^{-1}(\cdot)$ to represent the pre-image $\mathbf{S}$. In the formal description of our algorithms, we use the pre-image sampling algorithm $\mathsf{SamplePre}$.

zeros. Let us consider the case where the λ ciphertexts are encryptions of s. At a high level, the testing algorithm combines the ciphertext components appropriately, such that for each $i \leq \mathsf{nbp}$, the result is $\mathbf{B}^{(i)}_{L,\mathsf{rej}^{(i)}}$ if $\mathrm{PRF}(s,i) = 0$, and $\mathbf{B}^{(i)}_{L,\mathsf{acc}^{(i)}}$ otherwise. Once the testing algorithm gets these matrices, it can sum them to check if it is (close to) the zero matrix. The testing algorithm essentially mimics the program evaluation on s using the encryption of s. Let us fix a program $\mathsf{BP}^{(i)}$, and say it reads bit positions $p_1, \ldots, p_L$. At step 1, the program goes from state 1 at level 0 to state $\mathsf{st}_1 = \sigma^{(i)}_{1,s_{p_1}}$ at level 1. The test algorithm has $\mathbf{B}^{(i)}_{0,1}$. It combines this with the $(i,1)$ component of the p_1^{th} ciphertext to get $\mathbf{B}^{(i)}_{1,\mathsf{st}_1}$. Next, the program reads the bit at position p_2 and goes to state st_2 at level 2. The test algorithm, accordingly, combines $\mathbf{B}^{(i)}_{1,\mathsf{st}_1}$ with the $(i,2)$ sub-sub-component of the p_2^{th} ciphertext to compute $\mathbf{B}^{(i)}_{2,p_2}$. Proceeding this way, the actual program evaluation reaches either $\mathsf{acc}^{(i)}$ or $\mathsf{rej}^{(i)}$, and the test algorithm accordingly reaches either $\mathbf{B}^{(i)}_{L,\mathsf{acc}^{(i)}}$ or $\mathbf{B}^{(i)}_{L,\mathsf{rej}^{(i)}}$.

The solution described above, however, is not IND-CPA secure. To hide the encrypted bit without affecting the above computation, we will have to add some noise to each sub-sub-ciphertext. In particular, instead of outputting $\mathbf{B}_j^{(i)^{-1}}(\mathbf{D})$ for some matrix $\mathbf{D}$, we will now have $\mathbf{B}_j^{(i)^{-1}}(\mathbf{S} \cdot \mathbf{D} + \mathsf{noise})$,[3] where $\mathbf{S}$ is a low norm matrix. To prove IND-CPA security, we first switch the top level matrices to uniformly random matrices. Once we've done that, we can use LWE, together with the properties of lattice trapdoors, to argue that the top level sub-sub-ciphertexts look like random matrices from a low-norm distribution. As a result, we don't need trapdoors for the matrices at level $L-1$, and hence, they can be switched to uniformly random matrices. Using LWE with trapdoor properties, we can then switch the sub-sub-ciphertexts at level $L-1$ to random matrices. Proceeding this way, all sub-sub-ciphertexts can be made random Gaussian matrices. This concludes our proof.

Separation from Chosen Ciphertext Security One interesting question is whether achieving chosen ciphertext security (as opposed to IND-CPA security) makes a bit encryption system more likely to be resistant to circular security attacks. Here we show generically that achieving a bit encryption system that is IND-CCA secure, but not circular secure is no more difficult than our original separation problem. In particular, we show generically how to combine a IND-CPA secure, but not circular secure bit encryption with multi-bit CCA secure encryption to achieve a single bit encryption system that is IND-CPA secure. We note that Rothblum addressed CCA security, but used the more specific assumption of trapdoor permutations to achieve NIZKs.

Our transformation is fairly simple and follows in a similar manner to how an analogous theorem in Bishop, Hohenberger and Waters [10].

Relation to GGH15 Graph Based Multilinear Maps Our counterexample construction bears some similarities to the graph-induced multilinear maps scheme of Gentry, Gorbunov and Halevi [23]. In a graph induced multilinear maps scheme, we have an underlying graph G, and encodings of elements are relative to pairs of connected nodes in in the graphs. Given encodings of s_1 and s_2 relative to connected nodes $u \rightsquigarrow v$, one can compute an encoding of $s_1 + s_2$ relative to $u \rightsquigarrow v$. Similarly, given an encoding of s_1 relative to $u \rightsquigarrow v$ and an encoding of s_2 relative to $v \rightsquigarrow w$, one can compute an encoding of $s_1 \cdot s_2$ relative to $u \rightsquigarrow w$. Finally, one is allowed to zero-test corresponding to certain source-destination pairs. Gentry et al. gave a lattice based construction for graph-induced encoding scheme, where each vertex u has an associated matrix $\mathbf{A}_u$ (together with a trapdoor T_u). The encoding of an element s corresponding to the edge (u,v) is simply $\mathbf{A}_u^{-1}(s\mathbf{A}_v + \mathsf{noise})$.

At a high level, our construction looks similar to the GGH15 multilinear maps construction. In particular, while GGH15 uses the cascading cancellations property to prove correctness, we use it for proving that the testing algorithm succeeds with high probability. Our security requirements, on the other hand, are

[3] Strictly speaking, if $\mathbf{D}$ consists of 5 components $\mathbf{D}_1, \ldots, \mathbf{D}_5$ stacked together, then our sub-sub-ciphertext will be $\mathbf{B}_j^{(i)^{-1}}(\mathbf{D'} + \mathsf{noise})$ where $\mathbf{D'}$ consists of 5 components $\mathbf{S} \cdot \mathbf{D_k}$ for $k \leq 5$.

different from that in multilinear maps. However, we believe that the ideas used in this work can be used to prove security of GGH15 mmaps for special graphs/secret distributions (note that GGH15 gave a candidate multilinear maps construction, and it did not have a proof of security for general graphs).

Summary and Conclusions To summarise, we show how to perform computation using an *outside primitive* by means of our oblivious sequence transformation approach. This allows us to show a separation between private-key semantic security and circular security for bit encryption schemes. While such counterexamples are contrived and do not give much insight into the circular security of existing schemes, we see this as a primitive of its own. The tools/techniques used for developing such counterexamples might have other applications. In particular, these counterexamples share certain features with more advanced cryptographic primitives such as witness encryption and code obfuscation.

2 Preliminaries

Notations. We will use lowercase bold letters for vectors (e.g. $\mathbf{v}$) and uppercase bold letters for matrices (e.g. $\mathbf{A}$). For any finite set S, $x \leftarrow S$ denotes a uniformly random element x from the set S. Similarly, for any distribution $\mathcal{D}$, $x \leftarrow \mathcal{D}$ denotes an element x drawn from distribution $\mathcal{D}$. The distribution $\mathcal{D}^n$ is used to represent a distribution over vectors of n components, where each component is drawn independently from the distribution $\mathcal{D}$.

Min-Entropy and Randomness Extraction. The min-entropy of a random variable X is defined as $\mathbf{H}_\infty(X) \stackrel{\text{def}}{=} -\log_2(\max_x \Pr[X = x])$. Let $\mathsf{SD}(X, Y)$ denote the statistical distance between two random variables X and Y. Below we state the Leftover Hash Lemma (LHL) from [21, 20].

Theorem 2.1. Let $\mathcal{H} = \{h : X \to Y\}_{h \in \mathcal{H}}$ be a universal hash family, then for any random variable W taking values in X, the following holds

$$\mathsf{SD}\left((h, h(W)), (h, U_Y)\right) \leq \frac{1}{2}\sqrt{2^{-\mathbf{H}_\infty(W)} \cdot |Y|}\ .$$

We will use the following corollary, which follows from the Leftover Hash Lemma.

Corollary 2.1. Let $\ell > m \cdot n \log_2 q + \omega(\log n)$ and q a prime. Let $\mathbf{R}$ be an $k \times m$ matrix chosen as per distribution $\mathcal{R}$, where $k = k(n)$ is polynomial in n and $\mathbf{H}_\infty(\mathcal{R}) = \ell$. Let $\mathbf{A}$ and $\mathbf{B}$ be matrices chosen uniformly in $\mathbb{Z}_q^{n \times k}$ and $\mathbb{Z}_q^{n \times m}$, respectively. Then the statistical distance between the following distributions is negligible in n.

$$\{(\mathbf{A}, \mathbf{A} \cdot \mathbf{R})\} \approx_s \{(\mathbf{A}, \mathbf{B})\}$$

Proof. The proof of above corollary follows directly from the Leftover Hash Lemma. Note that for a prime q the family of hash functions $h_{\mathbf{A}} : \mathbb{Z}_q^{k \times m} \to \mathbb{Z}_q^{n \times m}$ for $\mathbf{A} \in \mathbb{Z}_q^{n \times k}$ defined by $h_{\mathbf{A}}(\mathbf{X}) = \mathbf{A} \cdot \mathbf{X}$ is universal. Therefore, if $\mathcal{R}$ has sufficient min-entropy, i.e. $\ell > m \cdot n \log_2 q + \omega(\log n)$, then the Leftover Hash Lemma states that statistical distance between the distributions $(\mathbf{A}, \mathbf{A} \cdot \mathbf{R})$ and $(\mathbf{A}, \mathbf{B})$ is at most $2^{-\omega(\log n)}$ which is negligible in n as desired. ∎

2.1 Lattice Preliminaries

This section closely follows [26].

Given positive integers n, m, q and a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, we let $\Lambda_q^\perp(\mathbf{A})$ denote the lattice $\{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A} \cdot \mathbf{x} = \mathbf{0} \mod q\}$. For $\mathbf{u} \in \mathbb{Z}_q^n$, we let $\Lambda_q^{\mathbf{u}}(\mathbf{A})$ denote the coset $\{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A} \cdot \mathbf{x} = \mathbf{u} \mod q\}$.

Discrete Gaussians. Let σ be any positive real number. The Gaussian distribution $\mathcal{D}_\sigma$ with parameter σ is defined by the probability distribution function $\rho_\sigma(\mathbf{x}) = \exp(-\pi \cdot ||\mathbf{x}||^2/\sigma^2)$. For any set $\mathcal{L} \subset \mathcal{R}^m$, define $\rho_\sigma(\mathcal{L}) = \sum_{\mathbf{x}\in\mathcal{L}} \rho_\sigma(\mathbf{x})$. The discrete Gaussian distribution $\mathcal{D}_{\mathcal{L},\sigma}$ over $\mathcal{L}$ with parameter σ is defined by the probability distribution function $\rho_{\mathcal{L},\sigma}(\mathbf{x}) = \rho_\sigma(\mathbf{x})/\rho_\sigma(\mathcal{L})$ for all $\mathbf{x} \in \mathcal{L}$.

The following lemma (Lemma 4.4 of [31], [24]) shows that if the parameter σ of a discrete Gaussian distribution is small, then any vector drawn from this distribution will be short (with high probability).

Lemma 2.1. Let m, n, q be positive integers with $m > n$, $q \geq 2$. Let $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ be a matrix of dimensions $n \times m$, $\sigma = \tilde{\Omega}(n)$ and $\mathcal{L} = \Lambda_q^\perp(\mathbf{A})$. Then

$$\Pr[||\mathbf{x}|| > \sqrt{m} \cdot \sigma : \mathbf{x} \leftarrow \mathcal{D}_{\mathcal{L},\sigma}] \leq \text{negl}(n).$$

Learning with Errors (LWE). The Learning with Errors (LWE) problem was introduced by Regev [34]. The LWE problem has four parameters: the dimension of the lattice n, the number of samples m, the modulus q and the error distribution $\chi(n)$.

Assumption 1 (Learning with Errors)**.** Let n, m and q be positive integers and χ a noise distribution on $\mathbb{Z}$. The Learning with Errors assumption (n, m, q, χ)-LWE, parameterized by n, m, q, χ, states that the following distributions are computationally indistinguishable:

$$\left\{(\mathbf{A}, \mathbf{s}^\top \cdot \mathbf{A} + \mathbf{e}) \ : \ \begin{array}{l}\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m},\\ \mathbf{s} \leftarrow \mathbb{Z}_q^n, \mathbf{e} \leftarrow \chi^m\end{array}\right\} \approx_c \left\{(\mathbf{A}, \mathbf{u}) \ : \ \begin{array}{l}\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m},\\ \mathbf{u} \leftarrow \mathbb{Z}_q^m\end{array}\right\}$$

Under a quantum reduction, Regev [34] showed that for certain noise distributions, LWE is as hard as worst case lattice problems such as the decisional approximate shortest vector problem (GapSVP) and approximate shortest independent vectors problem (SIVP). The following theorem statement is from Peikert's survey [33].

Theorem 2.2 ([34])**.** For any $m \leq \mathsf{poly}(n)$, any $q \leq 2^{\mathsf{poly}(n)}$, and any discretized Gaussian error distribution χ of parameter $\alpha \cdot q \geq 2 \cdot \sqrt{n}$, solving (n, m, q, χ)-LWE is as hard as quantumly solving GapSVP_γ and SIVP_γ on arbitrary n-dimensional lattices, for some $\gamma = \tilde{O}(n/\alpha)$.

Later works [32, 15] showed classical reductions from LWE to GapSVP_γ. Given the current state of art in lattice algorithms, GapSVP_γ and SIVP_γ are believed to be hard for $\gamma = \tilde{O}(2^{n^\epsilon})$, and therefore (n, m, q, χ)-LWE is believed to be hard for Gaussian error distributions χ with parameter $2^{-n^\epsilon} \cdot q \cdot \mathsf{poly}(n)$.

LWE with Short Secrets. In this work, we will be using a variant of the LWE problem called *LWE with Short Secrets*. In this variant, introduced by Applebaum et al. [6], the secret vector is also chosen from the noise distribution χ. They showed that this variant is as hard as LWE for sufficiently large number of samples m.

Assumption 2 (LWE with Short Secrets)**.** Let n, m and q be positive integers and χ a noise distribution on $\mathbb{Z}$. The LWE with Short Secrets assumption (n, m, q, χ)-$\mathsf{LWE\text{-}ss}$, parameterized by n, m, q, χ, states that the following distributions are computationally indistinguishable [4]:

$$\left\{(\mathbf{A}, \mathbf{S} \cdot \mathbf{A} + \mathbf{E}) \ : \ \begin{array}{l}\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m},\\ \mathbf{S} \leftarrow \chi^{n\times n}, \mathbf{E} \leftarrow \chi^{n\times m}\end{array}\right\} \approx_c \left\{(\mathbf{A}, \mathbf{U}) \ : \ \begin{array}{l}\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m},\\ \mathbf{U} \leftarrow \mathbb{Z}_q^{n\times m}\end{array}\right\}.$$

[4] Applebaum et al. showed that $\{(\mathbf{A}, \mathbf{s}^\top \cdot \mathbf{A} + \mathbf{e}) : \mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}, \mathbf{s} \leftarrow \chi^n, \mathbf{e} \leftarrow \chi^m\} \approx_c \{(\mathbf{A}, \mathbf{u}) : \mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}, \mathbf{u} \leftarrow \mathbb{Z}_q^m\}$, assuming LWE is hard. However, by a simple hybrid argument, we can replace vectors $\mathbf{s}, \mathbf{e}, \mathbf{u}$ with matrices $\mathbf{S}, \mathbf{E}, \mathbf{U}$ of appropriate dimensions.

Lattices with Trapdoors. Lattices with trapdoors are lattices that are statistically indistinguishable from randomly chosen lattices, but have certain 'trapdoors' that allow efficient solutions to hard lattice problems.

Definition 2.1. A trapdoor lattice sampler consists of algorithms $\mathsf{TrapGen}$ and $\mathsf{SamplePre}$ with the following syntax and properties:

- $\mathsf{TrapGen}(1^n, 1^m, q) \to (\mathbf{A}, T_{\mathbf{A}})$: The lattice generation algorithm is a randomized algorithm that takes as input the matrix dimensions n, m, modulus q, and outputs a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ together with a trapdoor $T_{\mathbf{A}}$.
- $\mathsf{SamplePre}(\mathbf{A}, T_{\mathbf{A}}, \mathbf{u}, \sigma) \to \mathbf{s}$: The presampling algorithm takes as input a matrix $\mathbf{A}$, trapdoor $T_{\mathbf{A}}$, a vector $\mathbf{u} \in \mathbb{Z}_q^n$ and a parameter $\sigma \in \mathcal{R}$ (which determines the length of the output vectors). It outputs a vector $\mathbf{s} \in \mathbb{Z}_q^m$.

These algorithms must satisfy the following properties:

1. *Correct Presampling:* For all vectors $\mathbf{u}$, parameters σ, $(\mathbf{A}, T_{\mathbf{A}}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$, and $\mathbf{s} \leftarrow \mathsf{SamplePre}(\mathbf{A}, T_{\mathbf{A}}, \mathbf{u}, \sigma)$, $\mathbf{A} \cdot \mathbf{s} = \mathbf{u}$ and $\|\mathbf{s}\|_\infty \leq \sqrt{m} \cdot \sigma$.
2. *Well Distributedness of Matrix:* The following distributions are statistically indistinguishable:
$$\{\mathbf{A} : (\mathbf{A}, T_{\mathbf{A}}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)\} \approx_s \{\mathbf{A} : \mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}\}.$$
3. *Well Distributedness of Preimage:* For all $(\mathbf{A}, T_{\mathbf{A}}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$, if $\sigma = \omega(\sqrt{n \cdot \log q \cdot \log m})$, then the following distributions are statistically indistinguishable:
$$\{\mathbf{s} : \mathbf{u} \leftarrow \mathbb{Z}_q^n, \mathbf{s} \leftarrow \mathsf{SamplePre}(\mathbf{A}, T_{\mathbf{A}}, \mathbf{u}, \sigma)\} \approx_s \mathcal{D}_{\mathbb{Z}^m, \sigma}.$$

These properties are satisfied by the gadget-based trapdoor lattice sampler of [30].

2.2 Branching Programs

Branching programs are a model of computation used to capture space-bounded computations [12, 9]. In this work, we will be using a restricted notion called *permutation branching programs*.

Definition 2.2 (Permutation Branching Program)**.** A permutation branching program of length L, width w and input space $\{0,1\}^n$ consists of a sequence of $2L$ permutations $\sigma_{i,b} : [w] \to [w]$ for $1 \leq i \leq L, b \in \{0,1\}$, an input selection function $\mathsf{inp} : [L] \to [n]$, an accepting state $\mathsf{acc} \in [w]$ and a rejection state $\mathsf{rej} \in [w]$. The starting state st_0 is set to be 1 without loss of generality. The branching program evaluation on input $x \in \{0,1\}^n$ proceeds as follows:

- For $i = 1$ to L,
  - Let $\mathsf{pos} = \mathsf{inp}(i)$ and $b = x_{\mathsf{pos}}$. Compute $\mathsf{st}_i = \sigma_{i,b}(\mathsf{st}_{i-1})$.
- If $\mathsf{st}_L = \mathsf{acc}$, output 1. If $\mathsf{st}_L = \mathsf{rej}$, output 0, else output $\perp$.

In a remarkable result, Barrington [9] showed that any circuit of depth d can be simulated by a permutation branching program of width 5 and length 4^d.

Theorem 2.3 ([9])**.** For any boolean circuit C with input space $\{0,1\}^n$ and depth d, there exists a permutation branching program BP of width 5 and length 4^d such that for all inputs $x \in \{0,1\}^n$, $C(x) = \mathsf{BP}(x)$.

Looking ahead, the permutation property is crucial for our construction in Section 4. We will also require that the permutation branching program has a fixed input-selector function inp. In our construction, we will have multiple branching programs, and all of them must read the same input bit at any level $i \leq L$.

Definition 2.3. A permutation branching program with input space $\{0,1\}^n$ is said to have a fixed input-selector $\mathsf{inp}(\cdot)$ if for all $i \leq L$, $\mathsf{inp}(i) = i \bmod n$.

Any permutation branching program of length L and input space $\{0,1\}^n$ can be easily transformed to a fixed input-selector branching program of length nL. In this work, we only require that all branching programs share the same input selector function $\mathsf{inp}(\cdot)$. The input selector which satisfies $\mathsf{inp}(i) = i \bmod n$ is just one possibility, and we stick with it for simplicity.

2.3 Symmetric Key Encryption and Pseudorandom Functions

Symmetric Key Encryption. A symmetric key encryption scheme SKBE with message space $\mathcal{M}$ consists of algorithms Setup, Enc, Dec with the following syntax.

- $\mathsf{Setup}(1^\lambda) \to \mathsf{sk}$. The setup algorithm takes as input the security parameter and outputs secret key sk.
- $\mathsf{Enc}(\mathsf{sk}, m \in \mathcal{M}) \to \mathsf{ct}$. The encryption algorithm takes as input a secret key sk and a message $m \in \mathcal{M}$. It outputs a ciphertext ct.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to y \in \mathcal{M}$. The decryption algorithm takes as input a secret key sk, ciphertext ct and outputs a message $y \in \mathcal{M}$.

A symmetric key encryption scheme must satisfy correctness and $\mathsf{IND\text{-}CPA}$ security.

Correctness: For any security parameter λ, message $m \in \mathcal{M}$, $\mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda)$,

$$\Pr[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{sk}, m)) \neq m] < \mathrm{negl}(\lambda)$$

where the probability is over the random coins used during encryption and decryption.

Security: In this work, we will be using the $\mathsf{IND\text{-}CPA}$ security notion.

Definition 2.4. Let $\mathsf{SKBE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ be a symmetric key encryption scheme. The scheme is said to be $\mathsf{IND\text{-}CPA}$ secure if for all security parameters λ, all PPT adversaries $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{SKBE},\mathcal{A}}(\lambda) = |\Pr[\mathcal{A} \text{ wins the } \mathsf{IND\text{-}CPA} \text{ game }] - 1/2|$ is negligible in λ, where the $\mathsf{IND\text{-}CPA}$ experiment is defined below:

- The challenger chooses $\mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda)$, and bit $b \leftarrow \{0,1\}$.
- The adversary queries the challenger for encryptions of polynomially many messages $m_i \in \mathcal{M}$, and for each query m_i, the challenger sends ciphertext $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{sk}, m_i)$ to $\mathcal{A}$.
- The adversary sends two challenge messages m_0^*, m_1^* to the challenger. The challenger sends $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{sk}, m_b^*)$ to $\mathcal{A}$.
- Identical to the pre-challenge phase, the adversary makes polynomially many encryption queries and the challenger responds as before.
- $\mathcal{A}$ sends its guess b' and wins if $b = b'$.

Pseudorandom Functions. A family of keyed functions $\mathrm{PRF} = \{\mathrm{PRF}_\lambda\}_{\lambda \in \mathbb{N}}$ is a pseudorandom function family with key space $\mathcal{K} = \{\mathcal{K}_\lambda\}_{\lambda \in \mathbb{N}}$, domain $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$ and co-domain $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda \in \mathbb{N}}$ if function $\mathrm{PRF}_\lambda : \mathcal{K}_\lambda \times \mathcal{X}_\lambda \to \mathcal{Y}_\lambda$ is efficiently computable, and satisfies the pseudorandomness property defined below.

Definition 2.5. A pseudorandom function family PRF is secure if for every PPT adversary $\mathcal{A}$, there exists a negligible function $\mathrm{negl}(\cdot)$ such that

$$\left|\Pr[\mathcal{A}^{\mathrm{PRF}_\lambda(s,\cdot)}(1^\lambda) = 1] - \Pr[\mathcal{A}^{\mathcal{O}(\cdot)}(1^\lambda) = 1]\right| < \mathrm{negl}(\lambda),$$

where $\mathcal{O}$ is a random function and the probability is taken over the choice of seeds $s \in \mathcal{K}_\lambda$ and the random coins of the challenger and adversary.

Theorem 2.4. (PRFs in $\mathbf{NC}^1$ [7]) For some $\sigma > 0$, suitable universal constant $C > 0$, modulus $p \geq 2$, any $m = \mathsf{poly}(n)$, let $\chi = \mathcal{D}_{\mathbb{Z},\sigma}$ and $q \geq p \cdot k(C\sigma\sqrt{n})^k \cdot n^{\omega(1)}$, assuming hardness of (n, m, q, χ)-LWE, there exists a function family PRF consisting of functions from $\{0,1\}^k$ to $\mathbb{Z}_p^{m \times n}$ that satisfies pseudorandomness property as per Definition 2.5 and the entire function can be computed in $\mathbf{TC}^0 \subseteq \mathbf{NC}^1$.

From Theorems 2.3 and 2.4, the following corollary is immediate.

Corollary 2.2. Assuming hardness of (n, m, q, χ)-LWE with parameters as in Theorem 2.4, there exists a family of branching programs $\mathsf{BP} = \{\mathsf{BP}_\lambda\}_{\lambda \in \mathbb{N}}$ with input space $\{0,1\}^\lambda \times \{0,1\}^\lambda$ of width 5 and length $\mathsf{poly}(\lambda)$ that computes a pseudorandom function family.

3 Circular Security for Symmetric-Key Bit Encryption and Framework for Generating Separations

In this section, we define the notion of circular security for symmetric-key bit-encryption schemes. We also extend the BHW framework [10] to separate IND-CPA and circular security for bit-encryption in the symmetric-key setting. Informally, the circular security definition requires that it should be infeasible for any adversary to distinguish between encryption of the secret key and encryption of all-zeros string. In the bit-encryption case, each secret key bit is encrypted separately and independently.

Definition 3.1. (1-Circular Security for Bit Encryption) Let $\mathsf{SKBE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ be a symmetric-key bit-encryption scheme. Consider the following security game:

- The challenger chooses $\mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda)$ and $b \leftarrow \{0,1\}$.
- The adversary is allowed to make following queries polynomially many times:
  1. **Encryption Query.** It queries the challenger for encryption of message $m \in \{0,1\}$.
  2. **Secret Key Query.** It queries the challenger for encryption of i^{th} bit of the secret key sk.
- The challenger responds as follows:
  1. **Encryption Query.** For each query m, it computes the ciphertext $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{sk}, m)$, and sends ct to the adversary.
  2. **Secret Key Query.** For each query $i \leq |\mathsf{sk}|$, if $b = 0$, it sends the ciphertext $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{sk}, \mathsf{sk}_i)$, else it sends $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{sk}, 0)$.
- The adversary sends its guess b' and wins if $b = b'$.

The scheme SKBE is said to be circular secure if it satisfies semantic security (Definition 2.4), and for all security parameters λ, all PPT adversaries $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{bit\text{-}circ}}_{\mathsf{SKBE},\mathcal{A}}(\lambda) = |\Pr[\mathcal{A} \text{ wins}] - 1/2|$ is negligible in λ.

Next, we extend the BHW cycle tester framework for bit-encryption schemes.

3.1 Bit-Encryption Cycle Tester Framework

In a recent work, Bishop et al. [10] introduced a generic framework for separating IND-CPA and circular security. In their cycle tester framework, there are four algorithms - Setup, KeyGen, $\mathsf{Encrypt}$ and Test. The setup, key generation and encryption algorithms behave same as in any standard encryption scheme. However, the cycle tester does not contain a decryption algorithm, but provides a special testing algorithm. Informally, the testing algorithm takes as input a sequence of ciphertexts, and outputs 1 if the sequence corresponds to an encryption cycle, else it outputs 0. The security requirement is identical to semantic security for encryption schemes.

The BHW cycle tester framework is a useful framework for separating IND-CPA and n-circular security as it allows us to focus on building the core testing functionality without worrying about providing decryption. The full decryption capability is derived by generically combining a tester with a normal encryption scheme. The BHW framework does not directly work for generating circular security separations for bit-encryption. Below we provide a *bit-encryption cycle tester framework* for symmetric-key encryption along the lines of BHW framework.

Definition 3.2. (Bit-Encryption Cycle Tester) A symmetric-key cycle tester $\Gamma = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Test})$ for message space $\{0,1\}$ and secret key space $\{0,1\}^s$ is a tuple of algorithms (where $s = s(\lambda)$) specified as follows:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{sk}$. The setup algorithm takes as input the security parameter λ, and outputs a secret key $\mathsf{sk} \in \{0,1\}^s$.
- $\mathsf{Enc}(\mathsf{sk}, m \in \{0,1\}) \to \mathsf{ct}$. The encryption algorithm takes as input a secret key sk and a message $m \in \{0,1\}$, and outputs a ciphertext ct.
- $\mathsf{Test}(\mathbf{ct}) \to \{0,1\}$. The testing algorithm takes as input a sequence of s ciphertexts $\mathbf{ct} = (\mathsf{ct}_1, \ldots, \mathsf{ct}_s)$, and outputs a bit in $\{0,1\}$.

The algorithms must satisfy the following properties.

1. (Testing Correctness) There exists a polynomial $p(\cdot)$ such that for all security parameters λ, the Test algorithm's advantage in distinguishing sequence of encryptions of secret key bits from encryptions of zeros, denoted by $\mathsf{Adv}^{\mathsf{bit\text{-}circ}}_{\mathsf{SKBE},\mathsf{Test}}(\lambda)$ (Definition 3.1), is at least $1/p(\lambda)$.
2. (IND-CPA security) Let $\Pi = (\mathsf{Setup}, \mathsf{Enc}, \cdot)$ be an encryption scheme with empty decryption algorithm. The scheme Π must satisfy the IND-CPA security definition (Definition 2.4).

Next, we prove that given a cycle tester, we can transform any semantically secure bit-encryption scheme to another semantically secure bit-encryption scheme that is *circular insecure*.

3.2 Circular Security Separation from Cycle Testers

In this section, we prove the following theorem.

Theorem 3.1. (Separation from Cycle Testers) If there exists an IND-CPA secure symmetric-key bit-encryption scheme Π for message space $\{0,1\}$ and secret key space $\{0,1\}^{s_1}$ and symmetric-key bit-encryption cycle tester Γ for message space $\{0,1\}$ and secret key space $\{0,1\}^{s_2}$ (where $s_1 = s_1(\lambda)$ and $s_2 = s_2(\lambda)$), then there exists an IND-CPA secure symmetric-key bit-encryption scheme Π' for message space $\{0,1\}$ and secret key space $\{0,1\}^{s_1+s_2}$ that is circular insecure.

The proof of the above theorem is similar to that provided in [10], therefore we only provide a sketch.

Proof Sketch. Let $\Pi = (\mathsf{Setup}_1, \mathsf{Enc}_1, \mathsf{Dec}_1)$ and $\Gamma = (\mathsf{Setup}_2, \mathsf{Enc}_2, \mathsf{Test}_2)$. The idea is to use a normal encryption scheme and a cycle tester in parallel. The plaintext is encrypted under both schemes independently, and job of the normal encryption scheme shall be to allow decryption and cycle testing component guarantees cycle testability. Below, we construct an IND-CPA bit-encryption scheme $\Pi' = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$, together with a cycle testing algorithm Test, as follows.

- $\mathsf{Setup}(1^\lambda) \to \mathsf{sk}$. The setup algorithm takes as input the security parameter λ. It runs the setup algorithms as $\mathsf{sk}_1 \leftarrow \mathsf{Setup}_1(1^\lambda)$ and $\mathsf{sk}_2 \leftarrow \mathsf{Setup}_2(1^\lambda)$. It outputs the secret key $\mathsf{sk} = (\mathsf{sk}_1, \mathsf{sk}_2)$.
- $\mathsf{Enc}(\mathsf{sk}, m \in \{0,1\}) \to \mathsf{ct}$. The encryption algorithm takes as input a secret key $\mathsf{sk} = (\mathsf{sk}_1, \mathsf{sk}_2)$ and a message $m \in \{0,1\}$. It computes ciphertexts as $\mathsf{ct}_1 \leftarrow \mathsf{Enc}_1(\mathsf{sk}_1, m)$ and $\mathsf{ct}_2 \leftarrow \mathsf{Enc}_2(\mathsf{sk}_2, m)$. It outputs the ciphertext $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$.

- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to \{0,1\}$. The decryption algorithm takes as input secret key $\mathsf{sk} = (\mathsf{sk}_1, \mathsf{sk}_2)$ and ciphertext $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$. It decrypts the message as $m' = \mathsf{Dec}_1(\mathsf{sk}_1, \mathsf{ct}_1)$, and outputs m'.
- $\mathsf{Test}(\mathbf{ct}) \to \{0,1\}$. The testing algorithm takes as input a sequence of $s_1 + s_2$ ciphertexts $\mathbf{ct} = (\mathsf{ct}_1, \ldots, \mathsf{ct}_{s_1+s_2})$. It parses last s_2 ciphertexts as $\mathsf{ct}_j = (\mathsf{ct}_{j,1}, \mathsf{ct}_{j,2})$ for $j \geq s_1 + 1$, and outputs the bit $\mathsf{Test}_2(\mathbf{ct'})$, where $\mathbf{ct'} = (\mathsf{ct}_{s_1+1,2}, \ldots, \mathsf{ct}_{s_1+s_2,2})$.

The correctness of Dec and Test follow directly from that of Dec_1 and Test_2 (respectively). First, note that if a bit was correctly encrypted using Enc (and thereby correctly encrypted using Enc_1), then Dec decrypts the corresponding ciphertext to the same bit as it uses algorithm Dec_1. Second, if $\mathbf{ct}$ is a sequence of encryption of secret key bits, then its last s_2 ciphertexts must be encryptions of sk_2. Therefore, running Test_2 (on second half of last s_2 ciphertexts) should correctly distinguish encryption of secret key bits from encryption of zeros with non-negligible probability.

To complete the proof, we need to show that Π' is a semantically secure bit-encryption scheme (Definition 2.4). This follows from a simple hybrid argument in which we exploit the fact that each ciphertext generated using Enc is a pair of ciphertexts, where each ciphertext is generated using two different semantically secure bit-encryption schemes Π and Γ. So, in one hybrid step, we indistinguishably switch first part of challenge ciphertext from encryption of 0 to 1, and in the next step, we switch the second part of challenge ciphertext as well. ■

4 Private Key Bit-Encryption Cycle Tester

In this section, we present our Bit-Encryption Cycle Tester $\mathcal{E} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Test})$ satisfying Definition 3.2. Before describing the formal construction, we will give an outline of our construction and describe intuitively how the cycle testing algorithm works.

Outline of Our Construction: To begin with, let us first discuss the tools required for our bit-encryption cycle tester. The central primitive in our construction is a low depth pseudorandom function family. More specifically, we require a pseudorandom function $\mathrm{PRF} : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}$ (the first input is the PRF key, and the second input is the PRF input) such that for all $i < 2^\lambda$, $\mathrm{PRF}(\cdot, i)$[5] can be computed using a permutation branching program of polynomial length and polynomial width. Recall, from Corollary 2.2, there exist PRF constructions [7] that satisfy this requirement. Let $\mathsf{BP}^{(i)}$ denote a branching program of length L and width w computing $\mathrm{PRF}(\cdot, i)$. Each program $\mathsf{BP}^{(i)}$ has an accept state $\mathsf{acc}^{(i)} \in [w]$ and a reject state $\mathsf{rej}^{(i)} \in [w]$. We will also require that at each level $j \leq L$, all branching programs $\mathsf{BP}^{(i)}$ read the same input bit.

The setup algorithm first chooses the LWE parameters: the matrix dimensions n, m, LWE modulus q and noise χ. It also chooses a parameter nbp which is sufficiently larger than n, m and denotes the number of branching programs. Next, it chooses a PRF key s. Finally, for each state of each branching program, it chooses a 'random looking' matrix. In particular, it chooses matrices $\mathbf{B}^{(i)}_{j,k}$ for the state k at level j in $\mathsf{BP}^{(i)}$, and all these matrices have certain 'trapdoors'. The top level matrices corresponding to the accept/reject state satisfy a special constraint: for each branching program $\mathsf{BP}^{(i)}$, choose the matrix $\mathbf{B}^{(i)}_{L,\mathsf{acc}^{(i)}}$ if $\mathrm{PRF}(s, i) = 1$, else choose $\mathbf{B}^{(i)}_{L,\mathsf{rej}^{(i)}}$, and these chosen matrices must sum to 0. The secret key consists of the PRF key s and the matrices, together with their trapdoors.

Next, we describe the encryption algorithm. The ciphertexts are designed such that given an encryption of the secret key, we can combine the components appropriately in order to compute, for each $i \leq \mathsf{nbp}$, a noisy approximation of either $\mathbf{B}^{(i)}_{L,\mathsf{acc}^{(i)}}$ or $\mathbf{B}^{(i)}_{L,\mathsf{rej}^{(i)}}$ depending on $\mathrm{PRF}(s, i)$. If $\mathrm{PRF}(s, i) = 1$, then the output

[5]Here i is represented as a binary string of length λ

of this combination procedure is $\mathbf{B}^{(i)}_{L,\mathsf{acc}^{(i)}}$, else it is $\mathbf{B}^{(i)}_{L,\mathsf{rej}^{(i)}}$. As a result, adding these matrices results in the zero matrix. On the other hand, the same combination procedure with encryptions of zeroes gives us a matrix with large entries, thereby allowing us to break circular security. Let us now consider a simple case where we have two branching programs $\mathsf{BP}^{(1)}$, $\mathsf{BP}^{(2)}$, each of length $L = 4$, width $w = 3$ and reading two bit inputs (see Figure 1).

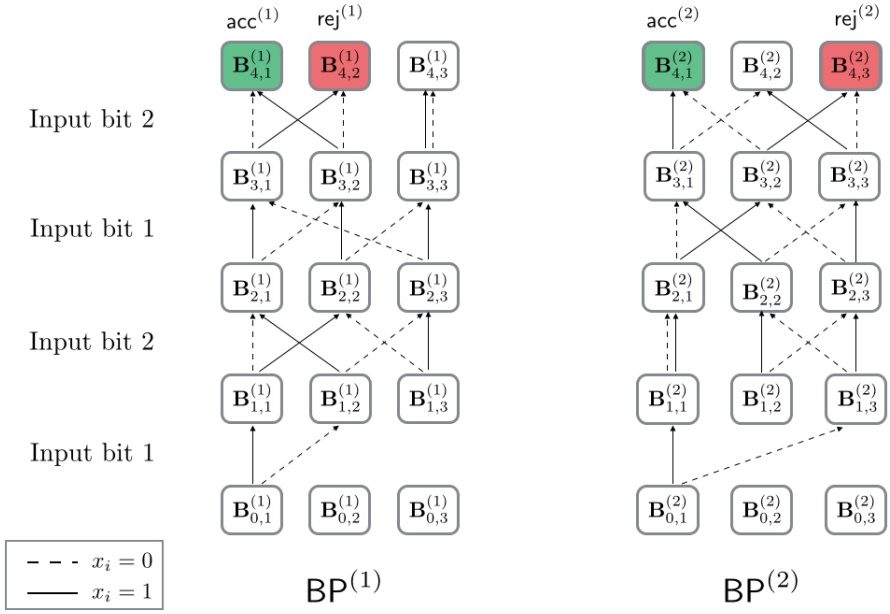

Figure 1: Branching programs $\mathsf{BP}^{(1)}$ and $\mathsf{BP}^{(2)}$.

Let us consider an encryption of a bit b. Each ciphertext consists of 4 sub-ciphertexts, one for each level. At each level, each sub-ciphertext consists of 2 sub-sub-ciphertexts, one for each branching program. The sub-sub-ciphertext $\mathsf{ct}^{(i)}_j$ at level j for program $\mathsf{BP}^{(i)}$ has the following 'propagation' property: for any state matrix $\mathbf{B}^{(i)}_{j-1,k}$ corresponding to state k at level $j-1$ in program $\mathsf{BP}^{(i)}$, $\mathbf{B}^{(i)}_{j-1,k} \cdot \mathsf{ct}^{(i)}_j = \mathbf{B}^{(i)}_{j,\sigma_b(k)}$. In our example (see Figure 1), if

$$\mathsf{ct} = \left(\left(\mathsf{ct}^{(1)}_1, \mathsf{ct}^{(2)}_1\right), \left(\mathsf{ct}^{(1)}_2, \mathsf{ct}^{(2)}_2\right), \left(\mathsf{ct}^{(1)}_3, \mathsf{ct}^{(2)}_3\right), \left(\mathsf{ct}^{(1)}_4, \mathsf{ct}^{(2)}_4\right)\right)$$

is an encryption of 0, then $\mathbf{B}^{(1)}_{2,3} \cdot \mathsf{ct}^{(1)}_3 = \mathbf{B}^{(1)}_{3,1}$. To achieve this, we use the lattice trapdoors. Finally, the ciphertext also contains the base level starting matrices $\{\mathbf{B}^{(i)}_{0,1}\}$.

To see how the test algorithm works, let us consider an encryption of the secret key. Recall, due to the cancellation property of the top level matrices, all we need is a means to compute $\mathbf{B}^{(i)}_{L,\mathsf{acc}^{(i)}}$ if $\mathsf{BP}^{(i)}(x) = 1$, else $\mathbf{B}^{(i)}_{L,\mathsf{rej}^{(i)}}$ if $\mathsf{BP}^{(i)}(x) = 0$. Let us consider $\mathsf{BP}^{(2)}$ in our example, and suppose we have encryptions $\mathsf{ct}[1]$ and $\mathsf{ct}[2]$ of bits 0 and 1 respectively. Now, from the propagation property, it follows that $\mathbf{B}^{(2)}_{0,1} \cdot \mathsf{ct}[1]^{(2)}_1 = \mathbf{B}^{(2)}_{1,3}$. Similarly, $\mathbf{B}^{(2)}_{1,3} \cdot \mathsf{ct}[2]^{(2)}_2 = \mathbf{B}^{(2)}_{2,3}$. Continuing this way, we can see that $\mathbf{B}^{(2)}_{0,1} \cdot \mathsf{ct}[1]^{(2)}_1 \cdot \mathsf{ct}[2]^{(2)}_2 \cdot \mathsf{ct}[1]^{(2)}_3 \cdot \mathsf{ct}[2]^{(2)}_4 = \mathbf{B}^{(2)}_{4,3}$. As a result, we have our desired $\mathbf{B}^2_{4,\mathsf{rej}^{(2)}}$. We can add the matrices computed for each $i \leq \mathsf{nbp}$, and see if they sum up to the zero matrix.

For proving security under LWE, we need to make some changes. Instead of having an exact propagation property, we will have an approximate version, where for any state matrix $\mathbf{B}^{(i)}_{j,k}$, $\mathbf{B}^{(i)}_{j,k} \cdot \mathsf{ct}^{(i)}_{j+1} \approx \mathbf{S}_{j+1} \cdot \mathbf{B}^{(i)}_{j+1,\sigma_b(k)}$.

Here $\mathbf{S}_{j+1}$ is a random low norm matrix chosen during encryption, and is common for all sub-sub-ciphertexts at level $j+1$. As a result, given an encryption of the secret key, at the top level, we either get an approximation of $\mathbf{T} \cdot \mathbf{B}^{(i)}_{L,\mathsf{acc}^{(i)}}$ or $\mathbf{T} \cdot \mathbf{B}^{(i)}_{L,\mathsf{rej}^{(i)}}$. Since $\mathbf{T}$ is a low norm matrix, adding the top-level outputs will be a low norm matrix if we have an encryption of the secret key.

4.1 Our Construction

Let $\mathrm{PRF} = \{\mathrm{PRF}_\lambda\}_{\lambda \in \mathbb{N}}$ be a family of secure pseudorandom functions, where $\mathrm{PRF}_\lambda : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}$ and for all $i \in \{0,1\}^\lambda$, $\mathrm{PRF}_\lambda(\cdot, i)$ can be computed by a fixed-input selector permutation branching program $\mathsf{BP}^{(i)}$ of length $L = \ell\text{-}\mathsf{bp}(\lambda)$ and width $w = \mathsf{w}\text{-}\mathsf{bp}(\lambda)$, where $\ell\text{-}\mathsf{bp}(\cdot)$ and $\mathsf{w}\text{-}\mathsf{bp}(\cdot)$ are fixed polynomials and

$$\mathsf{BP}^{(i)} = \left(\left\{ \sigma^{(i)}_{j,b} : [w] \to [w] \right\}_{j \in [L], b \in \{0,1\}}, \mathsf{acc}^{(i)} \in [w], \mathsf{rej}^{(i)} \in [w] \right).$$

Note that $\mathsf{BP}^{(i)}$ are fixed-input selector permutation branching programs, therefore they share the same input selector function $\mathsf{inp}(\cdot)$ defined as $\mathsf{inp}(i) = i \bmod n$ (see Definition 2.3). For simplicity of notation, we will drop the dependence on security parameter λ when it is clear from the context. Fix any $\epsilon < 1/2$. Below we describe our construction.

- $\mathsf{Setup}(1^\lambda) \to \mathsf{sk}$. The setup algorithm first chooses the following parameters: matrix dimensions n, m, LWE modulus q, parameter σ for the Gaussian noise distribution χ and an additional parameter nbp (which denotes the number of branching programs). Let $L = \ell\text{-}\mathsf{bp}(\lambda)$ and $w = \mathsf{w}\text{-}\mathsf{bp}(\lambda)$. Let $\mathsf{params} = (n, m, q, \sigma, \mathsf{nbp})$. The different parameters must satisfy the following constraints:
  - $n \geq \lambda$ (for LWE security)
  - $m = \Omega(n \cdot w \cdot \log q)$ (for $\mathsf{TrapGen}$)
  - $\chi = \mathcal{D}_{\mathbb{Z},\sigma}$ and $\sigma/q \geq \mathsf{poly}(n)/2^{n^\epsilon}$ (for LWE noise/modulus ratio to be greater than $\mathsf{poly}(n)/2^{n^\epsilon}$)
  - $\mathsf{nbp} \cdot L \cdot (m \cdot \sigma)^L < q/4$ (for the correctness of our Test algorithm)
  - $\mathsf{nbp} = \Omega\left(m \cdot n \cdot \log q\right)$ (for applying Leftover Hash Lemma)

  One possible setting of parameters is as follows: set n such that $w \cdot L \leq n^{\epsilon/2}$, $m = n \cdot w \cdot \log q \cdot \log n$, $\sigma = n^c$ for some constant c, $q = 2^{n^\epsilon}/n^c$ and $\mathsf{nbp} = m \cdot n \cdot \log q \cdot \log n$.

  Next, it chooses a random string $s \leftarrow \{0,1\}^\lambda$ and computes, for $i = 1$ to nbp, $t_i = \mathrm{PRF}(s, i)$.[6] It then samples $\mathsf{nbp} \cdot L$ matrices of dimensions $(w \cdot n) \times m$ along with their trapdoors (independently) as $(\mathbf{B}^{(i)}_j, T^{(i)}_j) \leftarrow \mathsf{TrapGen}(1^{w \cdot n}, 1^m, q)$ for $i = 1, \ldots, \mathsf{nbp}$ and $j = 0, \ldots, L-1$.

  It also chooses nbp uniformly random matrices $\mathbf{B}^{(i)}_L$ of dimensions $(w \cdot n) \times m$, such that the following constraint is satisfied

$$\sum_{i \,:\, t_i = 0} \mathbf{B}^{(i)}_{L,\mathsf{rej}^{(i)}} + \sum_{i \,:\, t_i = 1} \mathbf{B}^{(i)}_{L,\mathsf{acc}^{(i)}} = \mathbf{0}.$$

  Each matrix $\mathbf{B}^{(i)}_j \in \mathbb{Z}_q^{w \cdot n \times m}$ can be parsed as follows

$$\mathbf{B}^{(i)}_j = \begin{bmatrix} \mathbf{B}^{(i)}_{j,1} \\ \vdots \\ \mathbf{B}^{(i)}_{j,w} \end{bmatrix}$$

  where matrices $\mathbf{B}^{(i)}_{j,k} \in \mathbb{Z}_q^{n \times m}$ for $k \leq w$. Intuitively, the matrix $\mathbf{B}^{(i)}_{j,k}$ corresponds to state k at level j of branching program $\mathsf{BP}^{(i)}$.

[6] Here, i is represented as a λ bit string.

The algorithm sets secret key as $\mathsf{sk} = \left(s, \left\{\mathbf{B}_j^{(i)}, T_j^{(i)}\right\}_{i,j}, \mathsf{params}\right)$.

- $\mathsf{Encrypt}(\mathsf{sk}, m \in \{0,1\}) \to \mathsf{ct}$. The encryption algorithm takes as input the secret key sk and message m, where $\mathsf{sk} = \left(s, \left\{\mathbf{B}_j^{(i)}, T_j^{(i)}\right\}_{i,j}, \mathsf{params}\right)$. It runs the sub-encryption algorithm L times ($\mathsf{SubEncrypt}$ is defined in Figure 2) to compute L sub-ciphertexts.

  For $\mathsf{level} = 1$ to L, it computes the sub-ciphertexts at level level as

$$\mathsf{ct}_{\mathsf{level}} = \left(\mathsf{ct}_{\mathsf{level}}^{(1)}, \ldots, \mathsf{ct}_{\mathsf{level}}^{(\mathsf{nbp})}\right) \leftarrow \mathsf{SubEncrypt}(\mathsf{sk}, m, \mathsf{level}), \quad \forall\ \mathsf{level} \in \{1, \ldots, L\}\,.$$

SubEncrypt

Input: Secret key $\mathsf{sk} = (s, \{\mathbf{B}_j^{(i)}, T_j^{(i)}\}_{i \le \mathsf{nbp}, j \le L}, \mathsf{params})$, message $m \in \{0,1\}$, level $\mathsf{level} \in [L]$
Output: Sub-ciphertext $\mathsf{ct}_{\mathsf{level}}$.

1. Choose matrices $\mathbf{S} \leftarrow \chi^{n \times n}$ and $\mathbf{E}^{(i)} \leftarrow \chi^{w \cdot n \times m}$ for $i \le \mathsf{nbp}$.
2. Set matrix $\mathbf{D}^{(i)}$ as a permutation of the matrix blocks of $\mathbf{B}_{\mathsf{level}}^{(i)}$ according to the permutation $\sigma_{\mathsf{level},m}^{(i)}(\cdot)$. More formally, for $i \le \mathsf{nbp}$, set

$$\mathbf{D}^{(i)} = \begin{bmatrix} \mathbf{B}^{(i)}_{\mathsf{level}, \sigma^{(i)}_{\mathsf{level},m}(1)} \\ \vdots \\ \mathbf{B}^{(i)}_{\mathsf{level}, \sigma^{(i)}_{\mathsf{level},m}(w)} \end{bmatrix}.$$

3. Set $\mathbf{C}^{(i)} = (\mathbf{I}_w \otimes \mathbf{S}) \cdot \mathbf{D}^{(i)} + \mathbf{E}^{(i)}$ for $i \le \mathsf{nbp}$.
4. Compute $\mathsf{ct}^{(i)} \leftarrow \mathsf{SamplePre}(\mathbf{B}_{\mathsf{level}-1}^{(i)}, T_{\mathsf{level}-1}^{(i)}, \sigma, \mathbf{C}^{(i)})$ for $i \le \mathsf{nbp}$.
5. Output $\mathsf{ct}_{\mathsf{level}} = \left(\mathsf{ct}^{(1)}, \ldots, \mathsf{ct}^{(\mathsf{nbp})}\right)$.

Figure 2: Routine $\mathsf{SubEncrypt}$

Finally, it outputs the ciphertext as $\mathsf{ct} = \left(\left\{\mathbf{B}_{0,1}^{(i)}\right\}_i, \left\{\mathsf{ct}_j^{(i)}\right\}_{i,j}\right)$.

- $\mathsf{Test}(\mathsf{ct}[1], \ldots, \mathsf{ct}[\lambda], \ldots, \mathsf{ct}[|\mathsf{sk}|]) \to \{0,1\}$. The testing algorithm takes as input a sequence of $|\mathsf{sk}|$ ciphertexts $(\mathsf{ct}[1], \ldots, \mathsf{ct}[\lambda], \ldots)$. We will assume the algorithm also knows the LWE modulus q. It parses the first λ ciphertexts as $\mathsf{ct}[k] = \left(\left\{\mathbf{B}_{0,1}^{(i)}\right\}_i, \left\{\mathsf{ct}[k]_j^{(i)}\right\}_{i,j}\right)$ for $k \le \lambda$. Next, it computes the following

$$\mathsf{sum} = \sum_{i=1}^{\mathsf{nbp}} \mathbf{B}_{0,1}^{(i)} \cdot \prod_{j=1}^{L} \mathsf{ct}[\mathsf{inp}(j)]_j^{(i)}.$$

  If each component of sum lies in $(-q/4, q/4)$, then the algorithm outputs 1 to indicate a cycle. Otherwise it outputs 0. We would like to remind the reader that the starting state st_0 of each branching program $\mathsf{BP}^{(i)}$ is 1 (assumed w.l.o.g. in Section 2.2), therefore the testing algorithm only requires the matrices $\mathbf{B}_{0,1}^{(i)}$ to start oblivious evaluation of each branching program.

4.2 Proof of Correctness

In this section, we will prove correctness of our bit-encryption cycle tester. Concretely, we show that the Test algorithm distinguishes between a sequence of $|\mathsf{sk}|$ ciphertexts where k^{th} ciphertext encrypts k^{th} bit of the secret key, and a sequence of encryptions of zeros with non-negligible probability. First, we show that if Test algorithm is given encryptions of secret key bits, then it outputs 1 with all-but-negligible probability. Next, we show that if Test algorithm is run on encryptions of zeros, then it outputs 0 with all-but-negligible probability. Using these two facts, correctness of our cycle tester follows.

4.2.1 Testing Encryptions of Key Bits

Let $\mathbf{ct} = (\mathsf{ct}[1], \ldots, \mathsf{ct}[\lambda], \ldots)$ be the sequence of $|\mathsf{sk}|$ ciphertexts where k^{th} ciphertext encrypts bit sk_k, and it can be parsed as $\mathsf{ct}[k] = \left(\left\{ \mathbf{B}_{0,1}^{(i)} \right\}_i, \left\{ \mathsf{ct}[k]_j^{(i)} \right\}_{i,j} \right)$. Recall that the first λ bits of secret key sk correspond to the PRF key s. Therefore, $\mathsf{ct}[k]$ is an encryption of the bit s_k for $k \leq \lambda$. Also, i^{th} branching program $\mathsf{BP}^{(i)}$ computes the function $\mathrm{PRF}_\lambda(\cdot, i)$. This could be equivalently stated as

$$\forall\ i \leq \mathsf{nbp}, \qquad \sigma_{L,b_L}^{(i)} \left(\cdots \left(\sigma_{1,b_1}^{(i)}(1) \right) \cdots \right) = \begin{cases} \mathsf{rej}^{(i)} & \text{if } \mathrm{PRF}(s,i) = 0, \\ \mathsf{acc}^{(i)} & \text{if } \mathrm{PRF}(s,i) = 1 \end{cases}$$

where $b_j = s_{\mathsf{inp}(j)}$ for $j \leq L$. Let $\mathsf{st}_j^{(i)}$ denote the state of the i^{th} branching program after j steps. The initial state $\mathsf{st}_0^{(i)}$ is 1 for all programs, and j^{th} state can be computed as $\mathsf{st}_j^{(i)} = \sigma_{j,s_{\mathsf{inp}(j)}}^{(i)}(\mathsf{st}_{j-1}^{(i)})$.

Note that every ciphertext $\mathsf{ct}[k]$ consists of L sub-ciphertexts $\mathsf{ct}[k]_j$ for each level $j \leq L$, and each sub-ciphertext consists of nbp short matrices, each for a separate branching program. For constructing each sub-ciphertext, exactly one short secret matrix $\mathbf{S}_j$ is chosen, and it is shared across all nbp branching programs for generating LWE-type samples. It is crucial for testability that $\mathbf{S}_j$'s stay same for all branching programs.

First, we will introduce some notations for this proof.

- $\mathbf{S}[k]_j$: matrix chosen at level j for computing $\mathsf{ct}[k]_j^{(i)}$
- $\mathbf{E}[k]_j^{(i)}$: error matrix chosen at level j, program i for computing $\mathsf{ct}[k]_j^{(i)}$
- $\mathsf{inp}_j = \mathsf{inp}(j)$: the input bit read at level j of the branching program
- $\mathbf{S}_j = \mathbf{S}[\mathsf{inp}_j]_j$, $\mathbf{E}_j^{(i)} = \mathbf{E}[\mathsf{inp}_j]_j^{(i)}$, $\mathsf{CT}_j^{(i)} = \mathsf{ct}[\mathsf{inp}_j]_j^{(i)}$
- $\mathbf{\Gamma}_{j^*} = \prod_{j=1}^{j^*} \mathbf{S}_j$
- $\mathbf{\Delta}_{j^*}^{(i)} = \mathbf{B}_{0,1}^{(i)} \cdot \left(\prod_{j=1}^{j^*} \mathsf{CT}_j^{(i)} \right)$, $\quad \widetilde{\mathbf{\Delta}}_{j^*}^{(i)} = \mathbf{\Gamma}_{j^*} \cdot \mathbf{B}_{j^*,\mathsf{st}_{j^*}^{(i)}}^{(i)}$, $\quad \mathbf{Err}_{j^*}^{(i)} = \mathbf{\Delta}_{j^*}^{(i)} - \widetilde{\mathbf{\Delta}}_{j^*}^{(i)}$

The Test algorithm checks that $\left\| \sum_{i=1}^{\mathsf{nbp}} \mathbf{\Delta}_L^{(i)} \right\|_\infty < q/4$. Also, note that

$$\sum_{i=1}^{\mathsf{nbp}} \widetilde{\mathbf{\Delta}}_L^{(i)} = \sum_{i=1}^{\mathsf{nbp}} \mathbf{\Gamma}_L \cdot \mathbf{B}_{L,\mathsf{st}_L^{(i)}}^{(i)} = \mathbf{\Gamma}_L \cdot \sum_{i=1}^{\mathsf{nbp}} \mathbf{B}_{L,\mathsf{st}_L^{(i)}}^{(i)} = \mathbf{0}.$$

Thus, it would be sufficient to show that, with high probability, $\mathbf{Err}_L^{(i)} = \mathbf{\Delta}_L^{(i)} - \widetilde{\mathbf{\Delta}}_L^{(i)}$ is bounded. We will show that for all $i \leq \mathsf{nbp}$, $j^* \leq L$, $\mathbf{Err}_{j^*}^{(i)}$ is bounded.

Lemma 4.1. $\forall\ i \in \{1, \ldots, \mathsf{nbp}\}, j^* \in \{1, \ldots, L\}, \quad \left\| \mathbf{Err}_{j^*}^{(i)} \right\|_\infty \leq j^* \cdot (m \cdot \sigma)^{j^*}$ with overwhelming probability.

Proof. The above lemma is proven by induction over j^*, and all arguments hold irrespective of the value of i. Therefore, for simplicity of notation, we will drop the dependence on i. We will slightly abuse the notation and use $\mathbf{B}_{j,\sigma_{j,m}^{(i)}}^{(i)}$ to denote the following matrix.

$$\mathbf{B}_{j,\sigma_{j,m}^{(i)}}^{(i)} = \begin{bmatrix} \mathbf{B}_{j,\sigma_{j,m}^{(i)}(1)}^{(i)} \\ \vdots \\ \mathbf{B}_{j,\sigma_{j,m}^{(i)}(w)}^{(i)} \end{bmatrix}.$$

Before proceeding to our inductive proof, we would like to note the following fact.

Fact 4.1. For all $j \le L$, $\mathsf{CT}_j^{(i)} \leftarrow \mathsf{SamplePre}(\mathbf{B}_{j-1}^{(i)}, T_{j-1}^{(i)}, \sigma, \mathbf{C}_j^{(i)})$, where $\mathbf{C}_j^{(i)} = (\mathbf{I}_w \otimes \mathbf{S}_j) \cdot \mathbf{B}_{j,\sigma_{j,m}^{(i)}}^{(i)} + \mathbf{E}_j^{(i)}$ and $m = s_{\mathsf{inp}_j}$.

Base case ($j^* = 1$)**.** We know that $\mathbf{\Delta}_1 = \mathbf{B}_{0,1} \cdot (\mathsf{CT}_1)$. Therefore, using Fact 4.1, we can say that $\mathbf{\Delta}_1 = \mathbf{S}_1 \cdot \mathbf{B}_{1,\mathsf{st}_1} + \mathbf{E}_{1,1} = \widetilde{\mathbf{\Delta}}_1 + \mathbf{E}_{1,1}$. Note that $\mathbf{E}_{1,1}$ is an $n \times m$ submatrix consisting of first n rows of $\mathbf{E}_1$. Thus, we could write the following

$$\|\mathbf{Err}_1\|_\infty = \left\|\mathbf{\Delta}_1 - \widetilde{\mathbf{\Delta}}_1\right\|_\infty = \|\mathbf{E}_{1,1}\|_\infty \le m \cdot \sigma.$$

This completes the proof of base case. For the induction step, we assume that the above lemma holds for $j^* - 1$, and show that it holds for j^* as well.

Induction Step. We know that $\mathbf{\Delta}_{j^*} = \mathbf{\Delta}_{j^*-1} \cdot (\mathsf{CT}_{j^*})$. Also, $\mathbf{\Delta}_{j^*-1} = \widetilde{\mathbf{\Delta}}_{j^*-1} + \mathbf{Err}_{j^*-1}$. So, we could write the following

$$\begin{aligned}
\mathbf{\Delta}_{j^*} &= \widetilde{\mathbf{\Delta}}_{j^*-1} \cdot \mathsf{CT}_{j^*} + \mathbf{Err}_{j^*-1} \cdot \mathsf{CT}_{j^*} \\
&= \mathbf{\Gamma}_{j^*-1} \cdot \left(\mathbf{B}_{j^*-1,\mathsf{st}_{j^*-1}} \cdot \mathsf{CT}_{j^*}\right) + \mathbf{Err}_{j^*-1} \cdot \mathsf{CT}_{j^*} \\
&= \mathbf{\Gamma}_{j^*-1} \cdot \left(\mathbf{S}_{j^*} \cdot \mathbf{B}_{j^*,\mathsf{st}_{j^*}} + \mathbf{E}_{j^*,\mathsf{st}_{j^*-1}}\right) + \mathbf{Err}_{j^*-1} \cdot \mathsf{CT}_{j^*} \\
&= \widetilde{\mathbf{\Delta}}_{j^*}^{(i)} + \mathbf{\Gamma}_{j^*-1} \cdot \mathbf{E}_{j^*,\mathsf{st}_{j^*-1}} + \mathbf{Err}_{j^*-1} \cdot \mathsf{CT}_{j^*}
\end{aligned}$$

Here, $\mathbf{E}_{j^*,\mathsf{st}_{j^*-1}}$ is an $n \times m$ submatrix of $\mathbf{E}_{j^*}$. Finally, we can bound $\mathbf{Err}_{j^*}$ as follows

$$\begin{aligned}
\|\mathbf{Err}_{j^*}\|_\infty = \left\|\mathbf{\Delta}_{j^*} - \widetilde{\mathbf{\Delta}}_{j^*}\right\|_\infty &= \left\|\mathbf{\Gamma}_{j^*-1} \cdot \mathbf{E}_{j^*,\mathsf{st}_{j^*-1}} + \mathbf{Err}_{j^*-1} \cdot \mathsf{CT}_{j^*}\right\|_\infty \\
&\le \left\|\mathbf{\Gamma}_{j^*-1} \cdot \mathbf{E}_{j^*,\mathsf{st}_{j^*-1}}\right\|_\infty + \|\mathbf{Err}_{j^*-1} \cdot \mathsf{CT}_{j^*}\|_\infty \\
&\le (n \cdot \sigma)^{j^*-1} \cdot m \cdot \sigma + (j^* - 1) \cdot (m \cdot \sigma)^{j^*-1} \cdot m \cdot \sigma \le j^* \cdot (m \cdot \sigma)^{j^*}
\end{aligned}$$

This completes the proof. ∎

Using Lemma 4.1, we can claim that for all $i \le \mathsf{nbp}$, $\left\|\mathbf{\Delta}_L^{(i)} - \widetilde{\mathbf{\Delta}}_L^{(i)}\right\|_\infty \le L \cdot (m \cdot \sigma)^L$. Therefore,

$$\|\mathsf{sum}\|_\infty = \left\|\sum_{i=1}^{\mathsf{nbp}} \mathbf{\Delta}_L^{(i)}\right\|_\infty = \left\|\sum_{i=1}^{\mathsf{nbp}} \mathbf{\Delta}_L^{(i)} - \sum_{i=1}^{\mathsf{nbp}} \widetilde{\mathbf{\Delta}}_L^{(i)}\right\|_\infty \le \mathsf{nbp} \cdot L \cdot (m \cdot \sigma)^L < q/4$$

Therefore, for our setting of parameters, if ciphertexts encrypt the secret key bit-by-bit, then Test algorithm outputs 1 with high probability.

4.2.2 Testing Encryptions of Zeros

Lemma 4.2. If PRF is a family of secure pseudorandom functions and challenge ciphertexts are encryptions of zeros, then Test outputs 0 with all-but-negligible probability.

Proof. Since the ciphertexts are encryptions of zeros, each branching program $\mathsf{BP}^{(i)}$ computes the the value $t_i' = \mathrm{PRF}_\lambda(0, i)$. Also, with high probability, t_i' and t_i can not be equal for all $i \le \lambda$ as otherwise PRF_λ will not be a secure pseudorandom function. Therefore, with high probability,

$$\widetilde{\mathsf{sum}} = \sum_{i=1}^{\mathsf{nbp}} \widetilde{\mathbf{\Delta}}_L^{(i)} = \left(\prod_{j=1}^{L} \mathbf{S}_j\right) \cdot \sum_{i=1}^{\mathsf{nbp}} \mathbf{B}_{L,\mathsf{st}_L^{(i)}}^{(i)} \ne \mathbf{0}.$$

Now, $\sum_{i=1}^{\mathsf{nbp}} \mathbf{B}^{(i)}_{L,\mathsf{st}_L^{(i)}}$ will be a uniformly random matrix in $\mathbb{Z}_q^{n\times m}$ as $t' \neq t$ and $\mathbf{B}^{(i)}_{L,\mathsf{st}_L^{(i)}}$ are randomly chosen for $i \leq \mathsf{nbp}$. Let $\mathbf{S}$ denote the product $\prod_{j=1}^{L} \mathbf{S}_j$ and $\mathbf{B}$ denote the sum $\sum_{i=1}^{\mathsf{nbp}} \mathbf{B}^{(i)}_{L,\mathsf{st}_L^{(i)}}$. We can write $\widetilde{\mathsf{sum}}$ as $\widetilde{\mathsf{sum}} = \mathbf{S} \cdot \mathbf{B}$, where $\mathbf{B}$ is a random $n \times m$ matrix. Thus, $\widetilde{\mathsf{sum}}$ is a random $n \times m$ matrix as $\mathbf{S}$, product of L full rank matrices, is also full rank. So, with high probability, at least one entry in matrix sum will have absolute value $> q/4$ which implies that Test outputs 0. ■

4.3 IND-CPA Proof

We will now show that the construction described above is $\mathsf{IND\text{-}CPA}$ secure. The adversary queries for ciphertexts, and each ciphertext consists of $L \cdot \mathsf{nbp}$ sub-sub-ciphertexts. In our proof, we will gradually switch the sub-sub-ciphertexts to random low-norm (Gaussian) matrices, starting with the top-level sub-ciphertext and moving down. Once all sub-ciphertexts are switched to Gaussian matrices, the adversary has no information about the challenge message.

Our proof proceeds via a sequence of hybrid games. First, we switch the PRF evaluation to a truly random nbp bit string. Next, we switch the top level matrices to truly random matrices. This is possible since nbp is much larger than n, m, and as a result, we can use Leftover Hash Lemma. Once all top level matrices are truly random, we can make the top-level sub-sub-ciphertexts to be random low norm (Gaussian) matrices. This follows from the LWE security, together with the Property 3 of lattice trapdoors. Once the top level sub-sub-ciphertexts are Gaussian, we do not require the trapdoors at level $L-1$. As a result, we can choose uniformly random matrices at level $L-1$. This will allow us to switch the sub-sub-ciphertexts at level $L-1$ to Gaussian matrices. Proceeding this way, we can switch all sub-sub-ciphertexts to Gaussian matrices.

We will first define the sequence of hybrid games, and then show that they are computationally indistinguishable.

4.3.1 Sequence of Hybrid Games

$\mathsf{Game}\ 0$**:** This corresponds to the original security game.

- **Setup Phase**
  1. The challenger first chooses the LWE parameters n, m, q, σ, χ and nbp. Recall $L = \ell\text{-}\mathsf{bp}(\lambda)$ and $w = \mathsf{w\text{-}bp}(\lambda)$.
  2. Next, it chooses a uniformly random string $s \leftarrow \{0,1\}^\lambda$ and sets $t_i = \mathrm{PRF}(s, i)$ for $i \leq \mathsf{nbp}$.
  3. For $i = 1$ to nbp and $j = 0$ to $L-1$, it chooses $(\mathbf{B}_j^{(i)}, T_j^{(i)}) \leftarrow \mathsf{TrapGen}(1^{w\cdot n}, 1^m, q)$.
  4. It chooses nbp uniformly random matrices $\mathbf{B}_L^{(i)}$ of dimensions $w \cdot n \times m$, such that the following constraint is satisfied
  $$\sum_{i\ :\ t_i=0} \mathbf{B}^{(i)}_{L,\mathsf{rej}^{(i)}} + \sum_{i\ :\ t_i=1} \mathbf{B}^{(i)}_{L,\mathsf{acc}^{(i)}} = \mathbf{0}.$$
  5. Finally, the challenger sets $\mathsf{sk} = \left(s, \left\{\mathbf{B}_j^{(i)}, T_j^{(i)}\right\}_{i,j}\right)$.
- **Pre-Challenge Query Phase**
  1. The adversary requests polynomially many encryption queries. The challenger responds to each encryption query as follows.
  For $j = 1$ to L, the challenger computes $\mathsf{ct}_j \leftarrow \mathsf{SubEncrypt}(\mathsf{sk}, m, j)$ and sends $\mathsf{ct} = \left(\left\{\mathbf{B}^{(i)}_{0,1}\right\}_i, (\mathsf{ct}_1, \ldots, \mathsf{ct}_L)\right)$.
- **Challenge Phase** The challenger chooses a bit $b \leftarrow \{0,1\}$, and computes the challenge ciphertext identical to any pre-challenge query ciphertext for bit b.
- **Post-Challenge Query Phase** This is identical to the pre-challenge query phase.
- **Guess** The adversary finally sends the guess b', and wins if $b = b'$.

Game 1: This hybrid experiment is similar to the previous one, except that the string $t = (t_1, \ldots, t_{\mathsf{nbp}})$ is a uniformly random nbp bit string. Also, in place of the PRF key in the secret key, we have an empty string $\perp$. Note that this does not affect the encryption algorithm since it works oblivious to the PRF key (the PRF key is not used during encryption).

- **Setup Phase**
  1. The challenger first chooses the LWE parameters n, m, q, σ, χ and nbp. Recall $L = \ell\text{-}\mathsf{bp}(\lambda)$ and $w = \mathsf{w}\text{-}\mathsf{bp}(\lambda)$.
  2. Next, it chooses $t \leftarrow \{0,1\}^{\mathsf{nbp}}$.
  3. For $i = 1$ to nbp and $j = 0$ to $L-1$, it chooses $(\mathbf{B}_j^{(i)}, T_j^{(i)}) \leftarrow \mathsf{TrapGen}(1^{w \cdot n}, 1^m, q)$.
  4. It chooses nbp uniformly random matrices $\mathbf{B}_L^{(i)}$ of dimensions $w \cdot n \times m$, such that the following constraint is satisfied
$$\sum_{i \,:\, t_i = 0} \mathbf{B}^{(i)}_{L,\mathsf{rej}^{(i)}} + \sum_{i \,:\, t_i = 1} \mathbf{B}^{(i)}_{L,\mathsf{acc}^{(i)}} = \mathbf{0}.$$
  5. Finally, the challenger sets $\mathsf{sk} = \left(\perp, \left\{\mathbf{B}_j^{(i)}, T_j^{(i)}\right\}_{i,j}\right)$.
- **Pre-Challenge Query Phase**
  1. The adversary requests polynomially many encryption queries. The challenger responds to each encryption query as follows.
     For $j = 1$ to L, the challenger computes $\mathsf{ct}_j \leftarrow \mathsf{SubEncrypt}(\mathsf{sk}, m, j)$ and sends $\mathsf{ct} = \left(\left\{\mathbf{B}_{0,1}^{(i)}\right\}_i, (\mathsf{ct}_1, \ldots, \mathsf{ct}_L)\right)$.
- **Challenge Phase** The challenger chooses a bit $b \leftarrow \{0,1\}$, and computes the challenge ciphertext identical to any pre-challenge query ciphertext for bit b.
- **Post-Challenge Query Phase** This is identical to the pre-challenge query phase.
- **Guess** The adversary finally sends the guess b', and wins if $b = b'$.

Game 2: In this hybrid experiment, the challenger chooses the top-level matrices $\mathbf{B}_L^{(i)}$ uniformly at random.

- **Setup Phase**
  1. The challenger first chooses the LWE parameters n, m, q, σ, χ and nbp. Recall $L = \ell\text{-}\mathsf{bp}(\lambda)$ and $w = \mathsf{w}\text{-}\mathsf{bp}(\lambda)$.
  2. Next, it chooses $t \leftarrow \{0,1\}^{\mathsf{nbp}}$.
  3. For $i = 1$ to nbp and $j = 0$ to $L-1$, it chooses $(\mathbf{B}_j^{(i)}, T_j^{(i)}) \leftarrow \mathsf{TrapGen}(1^{w \cdot n}, 1^m, q)$.
  4. For $i = 1$ to nbp, it chooses uniformly random matrices $\mathbf{B}_L^{(i)} \leftarrow \mathbb{Z}_q^{w \cdot n \times m}$ of dimensions $w \cdot n \times m$.
  5. Finally, the challenger sets $\mathsf{sk} = \left(\perp, \left\{\mathbf{B}_j^{(i)}, T_j^{(i)}\right\}_{i,j}\right)$.
- **Pre-Challenge Query Phase**
  1. The adversary requests polynomially many encryption queries. The challenger responds to each encryption query as follows.
     For $j = 1$ to L, the challenger computes $\mathsf{ct}_j \leftarrow \mathsf{SubEncrypt}(\mathsf{sk}, m, j)$ and sends $\mathsf{ct} = \left(\left\{\mathbf{B}_{0,1}^{(i)}\right\}_i, (\mathsf{ct}_1, \ldots, \mathsf{ct}_L)\right)$.
- **Challenge Phase** The challenger chooses a bit $b \leftarrow \{0,1\}$, and computes the challenge ciphertext identical to any pre-challenge query ciphertext for bit b.
- **Post-Challenge Query Phase** This is identical to the pre-challenge query phase.
- **Guess** The adversary finally sends the guess b', and wins if $b = b'$.

Next, we have a sequence of $3L$ hybrid experiments $\mathsf{Game}\ 2.\mathsf{level}.\{1,2,3\}$ for $\mathsf{level} = L$ to 1.

Game 2.level.1**:** In hybrids Game 2.level.1, the sub-ciphertexts corresponding to levels greater than level are Gaussian matrices. At level level, the sub-ciphertext computation does not use SubEncrypt routine. Instead, it chooses a uniformly random matrix and computes the SamplePre of the uniformly random matrix. Also, for levels greater than level -1, matrices $\mathbf{B}_j^{(i)}$ are chosen uniformly at random instead of being sampled using TrapGen.

- **Setup Phase**
  1. The challenger first chooses the LWE parameters n, m, q, σ, χ and nbp. Recall $L = \ell\text{-bp}(\lambda)$ and $w = \text{w-bp}(\lambda)$.
  2. Next, it chooses $t \leftarrow \{0,1\}^{\mathsf{nbp}}$.
  3. For $i = 1$ to nbp and $j = 0$ to level $- 1$, it chooses $(\mathbf{B}_j^{(i)}, T_j^{(i)}) \leftarrow \mathsf{TrapGen}(1^{w \cdot n}, 1^m, q)$.
  4. For $i = 1$ to nbp and $j =$ level to L, it chooses uniformly random $\mathbf{B}_j^{(i)} \leftarrow \mathbb{Z}_q^{w \cdot n \times m}$ of dimensions $w \cdot n \times m$.
  5. Finally, the challenger sets $\mathsf{sk} = \left(\perp, \left\{\mathbf{B}_j^{(i)}, T_j^{(i)}\right\}_{i,j}\right)$.
- **Pre-Challenge Query Phase**
  1. The adversary requests polynomially many encryption queries. The challenger responds to each encryption query as follows.
  2. For $j = 1$ to level $- 1$, the challenger computes $\mathsf{ct}_j \leftarrow \mathsf{SubEncrypt}(\mathsf{sk}, m, j)$.
  3. For $j =$ level, the challenger chooses uniformly random matrix $\mathbf{C}_j^{(i)} \leftarrow \mathbb{Z}_q^{w \cdot n \times m}$ and sets $\mathsf{ct}_j^{(i)} \leftarrow \mathsf{SamplePre}(\mathbf{B}_{j-1}^{(i)}, T_{j-1}^{(i)}, \sigma, \mathbf{C}_j^{(i)})$. It sets $\mathsf{ct}_j = (\mathsf{ct}_j^{(1)}, \ldots, \mathsf{ct}_j^{(\mathsf{nbp})})$.
  4. For $i = 1$ to nbp and $j =$ level $+ 1$ to L, the challenger chooses $\mathsf{ct}_j^{(i)} \leftarrow \chi^{m \times m}$. It sets $\mathsf{ct}_j = (\mathsf{ct}_j^{(1)}, \ldots, \mathsf{ct}_j^{(\mathsf{nbp})})$.
  5. Finally, it sets $\mathsf{ct} = \left(\left\{\mathbf{B}_{0,1}^{(i)}\right\}_i, (\mathsf{ct}_1, \ldots, \mathsf{ct}_L)\right)$ and sends ct to the adversary.
- **Challenge Phase** The challenger chooses a bit $b \leftarrow \{0,1\}$, and computes the challenge ciphertext identical to any pre-challenge query ciphertext for bit b.
- **Post-Challenge Query Phase** This is identical to the pre-challenge query phase.
- **Guess** The adversary finally sends the guess b', and wins if $b = b'$.

Game 2.level.2**:** In hybrids Game 2.level.2, the sub-ciphertexts corresponding to levels greater than level $- 1$ are Gaussian matrices.

- **Setup Phase**
  1. The challenger first chooses the LWE parameters n, m, q, σ, χ and nbp. Recall $L = \ell\text{-bp}(\lambda)$ and $w = \text{w-bp}(\lambda)$.
  2. Next, it chooses $t \leftarrow \{0,1\}^{\mathsf{nbp}}$.
  3. For $i = 1$ to nbp and $j = 0$ to level $- 1$, it chooses $(\mathbf{B}_j^{(i)}, T_j^{(i)}) \leftarrow \mathsf{TrapGen}(1^{w \cdot n}, 1^m, q)$.
  4. For $i = 1$ to nbp and $j =$ level to L, it chooses uniformly random $\mathbf{B}_j^{(i)} \leftarrow \mathbb{Z}_q^{w \cdot n \times m}$ of dimensions $w \cdot n \times m$.
  5. Finally, the challenger sets $\mathsf{sk} = \left(\perp, \left\{\mathbf{B}_j^{(i)}, T_j^{(i)}\right\}_{i,j}\right)$.
- **Pre-Challenge Query Phase**
  1. The adversary requests polynomially many encryption queries. The challenger responds to each encryption query as follows.
  2. For $j = 1$ to level $- 1$, the challenger computes $\mathsf{ct}_j \leftarrow \mathsf{SubEncrypt}(\mathsf{sk}, m, j)$.
  3. For $i = 1$ to nbp and $j =$ level to L, the challenger chooses $\mathsf{ct}_j^{(i)} \leftarrow \chi^{m \times m}$. It sets $\mathsf{ct}_j = (\mathsf{ct}_j^{(1)}, \ldots, \mathsf{ct}_j^{(\mathsf{nbp})})$.

4. Finally, it sets $\mathsf{ct} = \left(\left\{\mathbf{B}_{0,1}^{(i)}\right\}_i, (\mathsf{ct}_1, \ldots, \mathsf{ct}_L)\right)$ and sends ct to the adversary.

- **Challenge Phase** The challenger chooses a bit $b \leftarrow \{0,1\}$, and computes the challenge ciphertext identical to any pre-challenge query ciphertext for bit b.
- **Post-Challenge Query Phase** This is identical to the pre-challenge query phase.
- **Guess** The adversary finally sends the guess b', and wins if $b = b'$.

Game 2.level.3**:** In hybrids Game 2.level.3, matrices $\mathbf{B}_j^{(i)}$ are chosen uniformly at random instead of being sampled using TrapGen for levels greater than $\mathsf{level} - 2$.

- **Setup Phase**
  1. The challenger first chooses the LWE parameters n, m, q, σ, χ and nbp. Recall $L = \ell\text{-}\mathsf{bp}(\lambda)$ and $w = \mathsf{w\text{-}bp}(\lambda)$.
  2. Next, it chooses $t \leftarrow \{0,1\}^{\mathsf{nbp}}$.
  3. For $i = 1$ to nbp and $j = 0$ to $\mathsf{level} - 2$, it chooses $(\mathbf{B}_j^{(i)}, T_j^{(i)}) \leftarrow \mathsf{TrapGen}(1^{w \cdot n}, 1^m, q)$.
  4. For $i = 1$ to nbp and $j = \mathsf{level} - 1$ to L, it chooses uniformly random $\mathbf{B}_j^{(i)} \leftarrow \mathbb{Z}_q^{w \cdot n \times m}$ of dimensions $w \cdot n \times m$.
  5. Finally, the challenger sets $\mathsf{sk} = \left(\perp, \left\{\mathbf{B}_j^{(i)}, T_j^{(i)}\right\}_{i,j}\right)$.
- **Pre-Challenge Query Phase**
  1. The adversary requests polynomially many encryption queries. The challenger responds to each encryption query as follows.
  2. For $j = 1$ to $\mathsf{level} - 1$, the challenger computes $\mathsf{ct}_j \leftarrow \mathsf{SubEncrypt}(\mathsf{sk}, m, j)$.
  3. For $i = 1$ to nbp and $j = \mathsf{level}$ to L, the challenger chooses $\mathsf{ct}_j^{(i)} \leftarrow \chi^{m \times m}$. It sets $\mathsf{ct}_j = (\mathsf{ct}_j^{(1)}, \ldots, \mathsf{ct}_j^{(\mathsf{nbp})})$.
  4. Finally, it sets $\mathsf{ct} = \left(\left\{\mathbf{B}_{0,1}^{(i)}\right\}_i, (\mathsf{ct}_1, \ldots, \mathsf{ct}_L)\right)$ and sends ct to the adversary.
- **Challenge Phase** The challenger chooses a bit $b \leftarrow \{0,1\}$, and computes the challenge ciphertext identical to any pre-challenge query ciphertext for bit b.
- **Post-Challenge Query Phase** This is identical to the pre-challenge query phase.
- **Guess** The adversary finally sends the guess b', and wins if $b = b'$.

4.3.2 Indistinguishability of Hybrid Games

We now establish via a sequence of lemmas that no PPT adversary can distinguish between any two adjacent games with non-negligible advantage. To conclude, we show that the advantage of any PPT adversary in the last game is 0.

Let $\mathcal{A}$ be a PPT adversary that breaks the security of our construction in the IND-CPA security game (Definition 2.4). In Game i, advantage of $\mathcal{A}$ is defined as $\mathsf{Adv}_{\mathcal{A}}^i = |\Pr[\mathcal{A} \text{ wins}] - 1/2|$. We show via a sequence of claims that $\mathcal{A}$'s advantage is distinguishing between any two consecutive games must be negligible, otherwise there will be a poly-time attack on the security of some underlying primitive. Finally, in last game, we show that $\mathcal{A}$'s advantage in the last game is 0.

Lemma 4.3. If PRF is a family of secure pseudorandom functions, then for any PPT adversary $\mathcal{A}$, $|\mathsf{Adv}_{\mathcal{A}}^0 - \mathsf{Adv}_{\mathcal{A}}^1| \leq \mathrm{negl}(\lambda)$ for some negligible function $\mathrm{negl}(\cdot)$.

Proof. We describe a reduction algorithm $\mathcal{B}$ which plays the indistinguishability based game with PRF challenger. $\mathcal{B}$ runs the Setup Phase as in Game 0, except it does not choose a string $s \leftarrow \{0,1\}^\lambda$. $\mathcal{B}$ makes nbp queries to the PRF challenger, where in the i^{th} query it sends i to the PRF challenger and sets t_i as the challenger's response. $\mathcal{B}$ performs remaining steps as as in Game 0, and sends 1 to the PRF challenger if $\mathcal{A}$ guesses the bit correctly, otherwise it sends 0 to the PRF challenger as its guess.

Note that when PRF challenger honestly evaluates the PRF on each query, then $\mathcal{B}$ exactly simulates the view of $\mathsf{Game}\ 0$ for $\mathcal{A}$. Otherwise if PRF challenger behaves as a random function, then $\mathcal{B}$ exactly simulates the view of $\mathsf{Game}\ 1$. Therefore, if $|\mathsf{Adv}^0_{\mathcal{A}} - \mathsf{Adv}^1_{\mathcal{A}}|$ is non-negligible, then PRF is not secure pseudorandom function family. ∎

Lemma 4.4. For any adversary $\mathcal{A}$, $|\mathsf{Adv}^1_{\mathcal{A}} - \mathsf{Adv}^2_{\mathcal{A}}| \leq \mathrm{negl}(\lambda)$ for some negligible function $\mathrm{negl}(\cdot)$.

Proof. The proof of this lemma follows from Corollary 2.1 which itself follows from the Leftover Hash Lemma Theorem 2.1. Note that the difference between $\mathsf{Game}\ 1$ and 2 is the way top level matrices $\mathbf{B}^{(i)}_L$ are sampled during Setup Phase. In $\mathsf{Game}\ 1$, matrix $\mathbf{B}^{(\mathsf{nbp})}_{L,\mathsf{st}^{(i)}_L}$ is chosen as

$$\mathbf{B}^{(\mathsf{nbp})}_{L,\mathsf{st}^{(\mathsf{nbp})}_L} = -\left(\sum_{i\leq \mathsf{nbp}-1\ :\ t_i=0} \mathbf{B}^{(i)}_{L,\mathsf{rej}^{(i)}} + \sum_{i\leq \mathsf{nbp}-1\ :\ t_i=1} \mathbf{B}^{(i)}_{L,\mathsf{acc}^{(i)}}\right),$$

where $\mathsf{st}^{(\mathsf{nbp})}_L$ is $\mathsf{acc}^{(\mathsf{nbp})}$ if $t_{\mathsf{nbp}} = 1$, and $\mathsf{rej}^{(\mathsf{nbp})}$ otherwise. It can be equivalently written as follows

$$\mathbf{B}^{(\mathsf{nbp})}_{L,\mathsf{st}^{(\mathsf{nbp})}_L} = -\mathbf{A}\cdot\mathbf{R}, \qquad \mathbf{A} = \left[\mathbf{B}^{(1)}_{L,\mathsf{rej}^{(1)}} \,||\, \mathbf{B}^{(1)}_{L,\mathsf{acc}^{(1)}} \,||\, \ldots \,||\, \mathbf{B}^{(\mathsf{nbp}-1)}_{L,\mathsf{rej}^{(\mathsf{nbp}-1)}} \,||\, \mathbf{B}^{(\mathsf{nbp}-1)}_{L,\mathsf{acc}^{(\mathsf{nbp}-1)}}\right]$$

where $\mathbf{R} = \mathbf{u}\otimes\mathbf{I}_m \in \mathbb{Z}_q^{2m(\mathsf{nbp}-1)\times m}$, $\mathbf{u} = (u_1,\ldots,u_{2\mathsf{nbp}-2})^\top \in \{0,1\}^{2\mathsf{nbp}-2}$ and for all $i \leq \mathsf{nbp}-1$, $u_{2i} = t_i$ and $u_{2i-1} = 1-t_i$. That is, matrix $\mathbf{R}$ consists of $2\mathsf{nbp}-2$ submatrices where if $t_i = 1$, then its $2i^{th}$ submatrix is identity and $(2i-1)^{th}$ submatrix is zero, otherwise it is the opposite. Let $\mathcal{R}$ denote the distribution of matrix $\mathbf{R}$ as described above with t drawn uniformly from $\{0,1\}^{\mathsf{nbp}}$. Note that $\mathbf{H}_\infty(\mathcal{R}) = \mathsf{nbp}-1$ (min-entropy of $\mathcal{R}$), and $\mathsf{nbp} > m\cdot n\log_2 q + \omega(\log n)$. Therefore, it follows (from Corollary 2.1) that

$$\left\{\left(\mathbf{A}, \mathbf{B}^{(\mathsf{nbp})}_{L,\mathsf{st}^{(\mathsf{nbp})}_L} = -\mathbf{A}\cdot\mathbf{R}\right)\ :\ \mathbf{A}\leftarrow\mathbb{Z}_q^{n\times 2m(\mathsf{nbp}-1)}, \mathbf{R}\leftarrow\mathcal{R}\right\}$$
$$\approx_s$$
$$\left\{\left(\mathbf{A}, \mathbf{B}^{(\mathsf{nbp})}_{L,\mathsf{st}^{(\mathsf{nbp})}_L}\right)\ :\ \mathbf{A}\leftarrow\mathbb{Z}_q^{n\times 2m(\mathsf{nbp}-1)}, \mathbf{B}^{(\mathsf{nbp})}_{L,\mathsf{st}^{(\mathsf{nbp})}_L}\leftarrow\mathbb{Z}_q^{n\times m}\right\}$$

Thus, $|\mathsf{Adv}^1_{\mathcal{A}} - \mathsf{Adv}^2_{\mathcal{A}}|$ is negligible in the security parameter for all PPT adversaries $\mathcal{A}$. ∎

Lemma 4.5. If $(n, \mathsf{nbp}\cdot w\cdot m, q, \chi)$-$\mathsf{LWE\text{-}ss}$ assumption holds (Assumption 2), then for any PPT adversary $\mathcal{A}$, $|\mathsf{Adv}^2_{\mathcal{A}} - \mathsf{Adv}^{2.L.1}_{\mathcal{A}}| \leq \mathrm{negl}(\lambda)$ for some negligible function $\mathrm{negl}(\cdot)$.

Proof. The difference between $\mathsf{Game}\ 2$ and $2.L.1$ is the way top-level sub-ciphertexts (ct_L) are created for all encryption queries (including challenge query). Recall that ct_L contains nbp short matrices $\mathsf{ct}^{(i)}_L$, and each $\mathsf{ct}^{(i)}_L$ is sampled as $\mathsf{ct}^{(i)}_L \leftarrow \mathsf{SamplePre}(\mathbf{B}^{(i)}_{L-1}, T^{(i)}_{L-1}, \sigma, \mathbf{C}^{(i)}_L)$. In $\mathsf{Game}\ 2$, matrix $\mathbf{C}^{(i)}_L$ is computed as $\mathbf{C}^{(i)}_L = (\mathbf{I}_w\otimes\mathbf{S}_L)\cdot\mathbf{D}^{(i)}_L + \mathbf{E}^{(i)}_L$, where $\mathbf{D}^{(i)}_L$ is a permutation of $\mathbf{B}^{(i)}_L$ and $\mathbf{E}^{(i)}_L$ is chosen as $\mathbf{E}^{(i)}_L \leftarrow \chi^{w\cdot n\times m}$. On the other hand, in $\mathsf{Game}\ 2.L.1$, it is chosen as $\mathbf{C}^{(i)}_L \leftarrow \mathbb{Z}_q^{w\cdot n\times m}$.

For proving indistinguishability of $\mathsf{Game}\ 2$ and $2.L.1$, we need to sketch q intermediate hybrids, where q is the total number of queries made by $\mathcal{A}$.[7] In k^{th} hybrid, the challenger proceeds as $\mathsf{Game}\ 2.L.1$ while answering first k queries, and proceeds as in $\mathsf{Game}\ 2$ for answering remaining queries. Indistinguishability between any two consecutive intermediate hybrids follows directly from $\mathsf{LWE\text{-}ss}$ assumption. Below we describe a reduction algorithm $\mathcal{B}$ which plays the $\mathsf{LWE\text{-}ss}$ indistinguishability game.

[7] Here q includes the challenge query as well.

First, $\mathcal{B}$ receives as $\mathsf{LWE\text{-}ss}$ challenge two $n \times (\mathsf{nbp} \cdot w \cdot m)$ matrices $(\mathbf{F}, \mathbf{G})$. It parses $\mathbf{F}$ into nbp submatrices of dimensions $n \times (w \cdot m)$ as $[\mathbf{F}^{(1)} || \ldots || \mathbf{F}^{(\mathsf{nbp})}] = \mathbf{F}$. Further, each matrix $\mathbf{F}^{(i)}$ is parsed into w matrices of dimensions $n \times m$ as $[\mathbf{F}_1^{(i)} || \ldots || \mathbf{F}_w^{(i)}] = \mathbf{F}^{(i)}$. Similarly, it parses $\mathbf{G}$ as well. Next, it runs the Setup phase as in Game 2, except instead of choosing matrices $\mathbf{B}_L^{(i)}$ uniformly at random, it sets them as $\mathbf{B}_{L,v}^{(i)} = \mathbf{F}_v^{(i)}$ for $i \leq \mathsf{nbp}$ and $v \leq w$.

$\mathcal{B}$ answers the first $i-1$ queries as in Game $2.L.1$. On receiving k^{th} query m_k, it computes $L-1$ sub-ciphertexts ct_j $(j \leq L-1)$ honestly using sub-encrypt routine.[8] For computing sub-ciphertext ct_L, it first sets matrices $\mathbf{C}_L^{(i)}$ for $i \leq \mathsf{nbp}$ as follows

$$\mathbf{C}_L^{(i)} = \begin{bmatrix} \mathbf{G}^{(i)}_{\sigma^{(i)}_{L,m_k}(1)} \\ \vdots \\ \mathbf{G}^{(i)}_{\sigma^{(i)}_{L,m_k}(w)} \end{bmatrix}.$$

Next, it computes $\mathsf{ct}_L^{(i)} \leftarrow \mathsf{SamplePre}(\mathbf{B}_{L-1}^{(i)}, T_{L-1}^{(i)}, \sigma, \mathbf{C}_L^{(i)})$ for $i \leq \mathsf{nbp}$, and sets $\mathsf{ct}_L = (\mathsf{ct}_L^{(1)}, \ldots, \mathsf{ct}_L^{(\mathsf{nbp})})$. $\mathcal{B}$ answers k^{th} query as $\mathsf{ct} = (\mathsf{ct}_1, \ldots, \mathsf{ct}_L)$. Now, $\mathcal{B}$ answers remaining queries as in Game 2. Finally, $\mathcal{A}$ sends b' as its guess to $\mathcal{B}$. If $b = b'$, then $\mathcal{B}$ sends 1 to $\mathsf{LWE\text{-}ss}$ challenger to indicate that $\mathbf{G}$ consists of LWE samples, otherwise it sends 0.

Since, $\mathsf{LWE\text{-}ss}$ chooses $\mathbf{F}$ uniformly at random, therefore $\mathcal{B}$ simulates the distribution of $\mathbf{B}_L^{(i)}$ for $i \leq \mathsf{nbp}$ exactly. Next, if $\mathbf{G} = \mathbf{S} \cdot \mathbf{F} + \mathbf{E}$ for some matrices $\mathbf{S} \leftarrow \chi^{n \times n}$ and $\mathbf{E} \leftarrow \chi^{n \times (\mathsf{nbp} \cdot w \cdot m)}$, then $\mathcal{B}$ simulates the view of Game 2 for $\mathcal{A}$, otherwise it simulates the view of Game $2.L.1$. Therefore, if $|\mathsf{Adv}_{\mathcal{A}}^{2} - \mathsf{Adv}_{\mathcal{A}}^{2.L.1}|$ is non-negligible, then $\mathsf{LWE\text{-}ss}$ assumption does not hold. ∎

Lemma 4.6. If the *preimage well-distributedness property* of lattice trapdoor sampler $(\mathsf{TrapGen}, \mathsf{SamplePre})$ holds (Definition 2.1), then for every adversary $\mathcal{A}$, for any level $\mathsf{level} \in [L]$, $|\mathsf{Adv}_{\mathcal{A}}^{2.\mathsf{level}.1} - \mathsf{Adv}_{\mathcal{A}}^{2.\mathsf{level}.2}| \leq \mathrm{negl}(\lambda)$ for some negligible function $\mathrm{negl}(\cdot)$.

Proof. To prove indistinguishability of Game $2.\mathsf{level}.1$ and $2.\mathsf{level}.2$, we need to sketch q intermediate hybrids as in Lemma 4.5. In k^{th} intermediate hybrid, the challenger proceeds as Game $2.\mathsf{level}.2$ while answering first k queries, and proceeds as in Game $2.\mathsf{level}.1$ for answering remaining queries. Indistinguishability between any two consecutive intermediate hybrids follows from preimage well-distributedness property of lattice trapdoor sampler.

Observe that in $(k-1)^{th}$ intermediate hybrid, $\mathsf{ct}_{\mathsf{level}}^{(i)}$ is chosen as $\mathsf{ct}_{\mathsf{level}}^{(i)} \leftarrow \mathsf{SamplePre}(\mathbf{B}_{\mathsf{level}-1}^{(i)}, T_{\mathsf{level}-1}^{(i)}, \sigma, \mathbf{C}_{\mathsf{level}}^{(i)})$ for $i \leq \mathsf{nbp}$, where $\mathbf{C}_{\mathsf{level}}^{(i)} \leftarrow \mathbb{Z}_q^{w \cdot n \times m}$. On the other hand, in k^{th} intermediate hybrid, they are chosen as $\mathsf{ct}_{\mathsf{level}}^{(i)} \leftarrow \chi^{m \times m}$ for $i \leq \mathsf{nbp}$. By a simple hybrid argument, we can restate the preimage well-distributedness property for matrices instead of vectors such that for all $i \leq \mathsf{nbp}$, the following holds

$$\left\{ \mathsf{ct}_{\mathsf{level}}^{(i)} \;:\; \begin{array}{c} (\mathbf{B}_{\mathsf{level}-1}^{(i)}, T_{\mathsf{level}-1}^{(i)}) \leftarrow \mathsf{TrapGen}(1^{w \cdot n}, 1^m, q),\ \mathbf{C}_{\mathsf{level}}^{(i)} \leftarrow \mathbb{Z}_q^{w \cdot n \times m}, \\ \mathsf{ct}_{\mathsf{level}}^{(i)} \leftarrow \mathsf{SamplePre}(\mathbf{B}_{\mathsf{level}-1}^{(i)}, T_{\mathsf{level}-1}^{(i)}, \sigma, \mathbf{C}_{\mathsf{level}}^{(i)}) \end{array} \right\} \approx_s \left\{ \mathsf{ct}_{\mathsf{level}}^{(i)} \;:\; \mathsf{ct}_{\mathsf{level}}^{(i)} \leftarrow \chi^{w \cdot n \times m} \right\}.$$

Thus, by a hybrid argument over i, we can switch the nbp short matrices in sub-ciphertext $\mathsf{ct}_{\mathsf{level}}$ from being sampled using $\mathsf{SamplePre}$ to Gaussian matrices. Therefore, intermediate hybrid $k-1$ and k are statistically indistinguishable. Hence, using nbp intermediate hybrids between Game $2.\mathsf{level}.1$ and $2.\mathsf{level}.2$, we can switch level level sub-ciphertexts to low-norm Gaussian matrices for all queries such that if preimage well-distributedness property of lattice trapdoor sampler holds, then Game $2.\mathsf{level}.1$ and $2.\mathsf{level}.2$ are statistically indistinguishable for all $\mathsf{level} \leq L$. ∎

[8] If k^{th} query is the challenge query, then $m_k = b$. In other words, m_k will be the random challenge bit.

Lemma 4.7. If the *matrix well-distributedness property* of lattice trapdoor sampler $(\mathsf{TrapGen}, \mathsf{SamplePre})$ holds (Definition 2.1), then for every adversary $\mathcal{A}$, for any level $\mathsf{level} \in [L]$, $|\mathsf{Adv}_{\mathcal{A}}^{2.\mathsf{level}.2} - \mathsf{Adv}_{\mathcal{A}}^{2.\mathsf{level}.3}| \leq \mathrm{negl}(\lambda)$ for some negligible function $\mathrm{negl}(\cdot)$.

Proof. The proof of this lemma follows directly from the matrix well-distributedness property of lattice trapdoor sampler. First, note that in both Games 2.level.2 and 2.level.3 sub-ciphertexts at level level (for all queries) consist of nbp random low-norm Gaussian matrices. Thus, the challenger does not need to know the trapdoor of matrices at level $(\mathsf{level} - 1)$, that is matrices $\mathbf{B}_{\mathsf{level}-1}^{(i)}$ for all $i \leq \mathsf{nbp}$ can be sampled without trapdoor. The matrix well-distributedness property states that for all $i \leq \mathsf{nbp}$

$$\{\mathbf{B}_{\mathsf{level}-1}^{(i)} : (\mathbf{B}_{\mathsf{level}-1}^{(i)}, T_{\mathsf{level}-1}^{(i)}) \leftarrow \mathsf{TrapGen}(1^{w \cdot n}, 1^m, q)\} \approx_s \{\mathbf{B}_{\mathsf{level}-1}^{(i)} : \mathbf{B}_{\mathsf{level}-1}^{(i)} \leftarrow \mathbb{Z}_q^{w \cdot n \times m}\}.$$

Therefore, by a simple hybrid argument over i, we can move from Game 2.level.2 to 2.level.3 using matrix well-distributedness property of lattice trapdoor sampler with only negligible drop in the advantage. ∎

Lemma 4.8. If $(n, \mathsf{nbp} \cdot w \cdot m, q, \chi)$-LWE-ss assumption holds (Assumption 2), then for any PPT adversary $\mathcal{A}$, for any level $\mathsf{level} \in [L-1]$, $|\mathsf{Adv}_{\mathcal{A}}^{2.(\mathsf{level}+1).3} - \mathsf{Adv}_{\mathcal{A}}^{2.\mathsf{level}.1}| \leq \mathrm{negl}(\lambda)$ for some negligible function $\mathrm{negl}(\cdot)$.

Proof. The proof of this lemma is similar to that of Lemma 4.5. ∎

Lemma 4.9. For any PPT adversary $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{2.1.3} = 0$.

Proof. The proof of this lemma follows from the fact that in Game 2.1.3, each ciphertext contains $\mathsf{nbp}L$ random low-norm Gaussian matrices irrespective of the message bit being encrypted. Therefore, the distribution of ciphertexts when 0 is encrypted is identical to the distribution of ciphertexts when 1 is encrypted, thus they do not contain any information about the encrypted message bit. Hence, the advantage of any adversary is this game is exactly 0. ∎

References

[1] Tolga Acar, Mira Belenkiy, Mihir Bellare, and David Cash. Cryptographic agility and its relation to circular encryption. In *EUROCRYPT '10*, volume 6110 of LNCS, pages 403–422. Springer, 2010.

[2] Pedro Adão, Gergei Bana, Jonathan Herzog, and Andre Scedrov. Soundness and completeness of formal encryption: The cases of key cycles and partial information leakage. *Journal of Computer Security*, 17(5):737–797, 2009.

[3] Navid Alamati and Chris Peikert. Three's compromised too: Circular insecurity for any cycle length from (ring-)lwe. *IACR Cryptology ePrint Archive*, 2016:110, 2016.

[4] Jacob Alperin-Sheriff and Chris Peikert. Circular and KDM security for identity-based encryption. In *Public Key Cryptography*, pages 334–352, 2012.

[5] Benny Applebaum. Key-dependent message security: Generic amplification and completeness. In *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, pages 527–546, 2011.

[6] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *CRYPTO*, pages 595–618, 2009.

[7] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, pages 719–737, 2012.

[8] Boaz Barak, Iftach Haitner, Dennis Hofheinz, and Yuval Ishai. Bounded key-dependent message security. In *Advances in Cryptology - EUROCRYPT*, pages 423–444, 2010.

[9] D A Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc1. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, STOC '86, 1986.

[10] Allison Bishop, Susan Hohenberger, and Brent Waters. New circular security counterexamples from decision linear and learning with errors. In *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, pages 776–800, 2015.

[11] Dan Boneh, Shai Halevi, Michael Hamburg, and Rafail Ostrovsky. Circular-Secure Encryption from Decision Diffie-Hellman. In *CRYPTO '08*, volume 5157 of LNCS, pages 108–125, 2008.

[12] Allan Borodin, Danny Dolev, Faith E. Fich, and Wolfgang J. Paul. Bounds for width two branching programs. *SIAM J. Comput.*, 15(2):549–560, 1986.

[13] Zvika Brakerski and Shafi Goldwasser. Circular and leakage resilient public-key encryption under subgroup indistinguishability (or: Quadratic residuosity strikes back). *IACR Cryptology ePrint Archive*, 2010:226, 2010.

[14] Zvika Brakerski, Shafi Goldwasser, and Yael Tauman Kalai. Black-box circular-secure encryption beyond affine functions. In *Theory of Cryptography - 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Proceedings*, pages 201–218, 2011.

[15] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 575–584, 2013.

[16] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. *IACR Cryptology ePrint Archive*, 2001:19, 2001.

[17] David Cash, Matthew Green, and Susan Hohenberger. New definitions and separations for circular security. In *Public Key Cryptography - PKC*, pages 540–557, 2012.

[18] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, pages 3–12, 2015.

[19] Jean-Sébastien Coron, Craig Gentry, Shai Halevi, Tancrède Lepoint, Hemanta K. Maji, Eric Miles, Mariana Raykova, Amit Sahai, and Mehdi Tibouchi. Zeroizing without low-level zeroes: New MMAP attacks and their limitations. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, pages 247–266, 2015.

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

[21] Yevgeniy Dodis, Leonid Reyzin, and Adam D. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 523–540, 2004.

[22] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.

[23] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In *TCC*, 2015.

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

[25] Venkata Koppula, Kim Ramchen, and Brent Waters. Separations in circular security for arbitrary length key cycles. In *Theory of Cryptography Conference (TCC)*, 2015.

[26] Venkata Koppula and Brent Waters. Circular security counterexamples for arbitrary length cycles from LWE. *IACR Cryptology ePrint Archive*, 2016:117, 2016.

[27] Peeter Laud. Encryption cycles and two views of cryptography. In *NORDSEC 2002 - Proceedings of the 7th Nordic Workshop on Secure IT Systems (Karlstad University Studies 2002:31*, pages 85–100, 2002.

[28] Antonio Marcedone and Claudio Orlandi. Obfuscation ⇒ (IND-CPA security !⇒ circular security). In *Security and Cryptography for Networks - 9th International Conference, SCN 2014, Amalfi, Italy, September 3-5, 2014. Proceedings*, pages 77–90, 2014.

[29] Antonio Marcedone, Rafael Pass, and Abhi Shelat. Bounded kdm security from io and owf. In *SCN 2016*, 2016.

[30] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, pages 700–718, 2012.

[31] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM J. Comput.*, 37(1):267–302, April 2007.

[32] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 333–342, 2009.

[33] Chris Peikert. A decade of lattice cryptography. Cryptology ePrint Archive, Report 2015/939, 2015. `http://eprint.iacr.org/`.

[34] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 84–93, 2005.

[35] Ron Rothblum. On the circular security of bit-encryption. Cryptology ePrint Archive, Report 2012/102, 2012. `http://eprint.iacr.org/`.

A Separating Chosen Ciphertext Security from Circular Security

In this section, we construct a CCA-secure private key bit-encryption encryption scheme that is circular insecure from any CPA-secure private key bit-encryption encryption scheme which is circular insecure and CCA-secure private key (multi-bit) encryption scheme. Before describing our construction, we define the notion of IND-CCA security for private key encryption schemes, and of circular security for CCA-secure private key bit-encryption encryption schemes.

Definition A.1. Let $\mathsf{SKBE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ be a private key encryption scheme. The scheme is said to be IND-CCA secure if for all security parameters λ, all PPT adversaries $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{SKBE},\mathcal{A}}(\lambda) = |\Pr[\mathcal{A} \text{ wins the IND-CCA game}] - 1/2|$ is negligible in λ, where the IND-CCA experiment is defined below:

- **Setup Phase.** The challenger chooses $\mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda)$ and $b \leftarrow \{0,1\}$.
- **Pre-Challenge Phase.** The adversary is allowed to make following queries polynomially many times:
  1. **Encryption Query.** For each query m, it computes the ciphertext $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{sk}, m)$, and sends ct to the adversary.
  2. **Decryption Query.** For each query ct, it decrypts the ciphertext as $m \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$, and sends m to the adversary.
- **Challenge Phase.** The adversary sends two challenge messages m_0^*, m_1^* to the challenger. The challenger sends $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b^*)$ to $\mathcal{A}$.
- **Post-Challenge Phase.** Identical to the pre-challenge phase, except the adversary is not allowed to query challenge ciphertext ct^* for decryption.
- **Guess.** The adversary sends its guess b' and wins if $b = b'$.

Definition A.2. (1-Circular Security for CCA-secure Private Key Bit Encryption) Let $\mathsf{SKBE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ be a CCA-secure private key bit-encryption scheme. The scheme SKBE is said to be circular secure if it satisfies circular security (Definition 3.1) along with IND-CCA security (Definition A.1).

Now we show the main theorem that given a CPA-secure private key bit-encryption encryption scheme which is circular insecure, we can transform any CCA-secure private key (multi-bit) encryption scheme to another CCA-secure private key bit-encryption encryption scheme that is *circular insecure*. Below we state the formal theorem.

Theorem A.1. (Separating Chosen Ciphertext Security from Circular Security) If there exists an IND-CPA secure private key bit-encryption scheme Π_1 for message space $\{0,1\}$ and secret key space $\{0,1\}^{s_1}$ that is circular insecure, and an IND-CCA secure private key (multi-bit) encryption scheme Π_2 for message space $\{0,1\}^\ell$ and secret key space $\{0,1\}^{s_2}$ (where $s_1 = s_1(\lambda)$ and $s_2 = s_2(\lambda)$), then there exists an IND-CCA secure private key bit-encryption scheme Γ for message space $\{0,1\}$ and secret key space $\{0,1\}^{s_1+s_2}$ that is circular insecure.

Proof. Let $\Pi_1 = (\mathsf{Setup}_1, \mathsf{Enc}_1, \mathsf{Dec}_1)$ and $\Pi_2 = (\mathsf{Setup}_2, \mathsf{Enc}_2, \mathsf{Dec}_2)$. Let n be the number of random bits used by encryption algorithm Enc_1. We require that $\ell \geq n + 2$, i.e. the message space of Π_2 should be sufficiently large such that it can uniquely encode each random string in $\{0,1\}^n$ (random coin space of Enc_1) plus some 2-bit auxiliary information. Also, let us assume that Test_1 be the PPT cycle testing algorithm that takes as input a sequence of s_1 ciphertexts and tests whether they encrypt the secret key of scheme Π_1.

The idea is to encrypt the message bit under the IND-CPA secure scheme, and in parallel encrypt the same message bit along with randommess used during previous encryption under the IND-CCA secure scheme. The combined decryption algorithm first decrypts the second (IND-CCA) component of ciphertext and obtains the message bit and random string, and then it verifies the consistency of the first component by re-encrypting the message bit and random string under first private key. If the complete ciphertext is consistent, then it outputs the consistent message bit, else it outputs $\perp$. The cycle testability of the new scheme follows directly from the cycle testability of Π_1, and the IND-CCA security crucially relies on the fact that the ciphertext is decrypted if and only if both components of the ciphertext are consistent. In order for IND-CCA proof to completely go through the second component of each ciphertext needs to encrypt another bit which acts as a hidden trigger in the proof. Below, we describe our construction for an IND-CCA secure bit-encryption scheme $\Gamma = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$, together with a cycle testing algorithm Test.

- $\mathsf{Setup}(1^\lambda) \to \mathsf{sk}$. The setup algorithm takes as input the security parameter λ. It runs the setup algorithms as $\mathsf{sk}_1 \leftarrow \mathsf{Setup}_1(1^\lambda)$ and $\mathsf{sk}_2 \leftarrow \mathsf{Setup}_2(1^\lambda)$. It outputs the secret key as $\mathsf{sk} = (\mathsf{sk}_1, \mathsf{sk}_2)$.
- $\mathsf{Enc}(\mathsf{sk}, m \in \{0,1\}) \to \mathsf{ct}$. The encryption algorithm takes as input a secret key $\mathsf{sk} = (\mathsf{sk}_1, \mathsf{sk}_2)$ and a message $m \in \{0,1\}$. It chooses an n bit random string $r \leftarrow \{0,1\}^n$, and computes ciphertext ct_1 as $\mathsf{ct}_1 = \mathsf{Enc}_1(\mathsf{sk}_1, m; r)$, i.e. encrypts m under sk_1 using r as randommess. It also computes $\mathsf{ct}_2 \leftarrow \mathsf{Enc}_2(\mathsf{sk}_2, 0||m||r)$, and outputs the ciphertext $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to \{0,1\} \cup \perp$. The decryption algorithm takes as input secret key $\mathsf{sk} = (\mathsf{sk}_1, \mathsf{sk}_2)$ and ciphertext $\mathsf{ct} = (\mathsf{ct}_1, \mathsf{ct}_2)$. It first decrypts ct_2 to obtain $\alpha||m'||r' = \mathsf{Dec}_2(\mathsf{sk}_2, \mathsf{ct}_2)$. If $\alpha = 1$, then it outputs $\perp$. Else, it checks if $\mathsf{ct}_1 = \mathsf{Enc}_1(\mathsf{sk}_1, m'; r')$. If the check succeeds, then it outputs m', else it outputs $\perp$.
- $\mathsf{Test}(\mathbf{ct}) \to \{0,1\}$. The testing algorithm takes as input a sequence of $s_1 + s_2$ ciphertexts $\mathbf{ct} = (\mathsf{ct}_1, \ldots, \mathsf{ct}_{s_1+s_2})$. It parses the first s_1 ciphertexts as $\mathsf{ct}_j = (\mathsf{ct}_{j,1}, \mathsf{ct}_{j,2})$ for $j \leq s_1$, and outputs the bit $\mathsf{Test}_1(\mathbf{ct'})$, where $\mathbf{ct'} = (\mathsf{ct}_{1,1}, \ldots, \mathsf{ct}_{s_1,1})$.

The correctness of Dec follows directly from that of Dec_1, Dec_2 and Enc_1. Also, the correctness of cycle testing algorithm Test follows directly from that of Test_1.

First, note that if a bit m was correctly encrypted using Enc, i.e. it was correctly encrypted using Enc_1 with randommess r, and $(0||m||r)$ was correctly encrypted using Enc_2, then Dec decrypts ciphertext ct_2 to $(0||m||r)$ using Dec_2. Since the first bit of the plaintext is 0 and $\mathsf{ct}_1 = \mathsf{Enc}_1(\mathsf{pk}_1, m; r)$, therefore the consistency checks succeed and Dec outputs the same bit m.

Second, if $\mathbf{ct}$ is a sequence of encryption of secret key bits, then its first s_1 ciphertexts must be encryptions of sk_1. Therefore, running Test_1 (on first half of first s_2 ciphertexts) correctly distinguishes encryption of secret key bits from encryption of zeros with non-negligible probability. Thus, Test successfully tests a key cycle with non-negligible probability.

To complete the proof of Theorem A.1, we need to show that Γ satisfies IND-CCA security (Definition A.1). This follows from a simple hybrid argument. Let $\mathcal{A}$ be the PPT adversary playing the security game. Below we define the sequence of hybrid games, and later show that they are computationally indistinguishable.

Game 0**:** This corresponds to the original IND-CCA security game.

- **Setup Phase.** The challenger chooses secret key as $\mathsf{sk}_1 \leftarrow \mathsf{Setup}_1(1^\lambda)$, $\mathsf{sk}_2 \leftarrow \mathsf{Setup}_2(1^\lambda)$, bit $b \leftarrow \{0,1\}$. It sets $\mathsf{sk} = (\mathsf{sk}_1, \mathsf{sk}_2)$.
- **Pre-Challenge Phase.** The adversary is allowed to make following queries polynomially many times:
  1. **Encryption Query.** For each query m_i, the challenger chooses an n bit random string $r_i \leftarrow \{0,1\}^n$, and computes ciphertexts $\mathsf{ct}_{i,1}, \mathsf{ct}_{i,2}$ as $\mathsf{ct}_{i,1} = \mathsf{Enc}_1(\mathsf{sk}_1, m_i; r_i)$, $\mathsf{ct}_2 \leftarrow \mathsf{Enc}_2(\mathsf{sk}_2, 0||m_i||r_i)$, and sends the ciphertext $\mathsf{ct}_i = (\mathsf{ct}_{i,1}, \mathsf{ct}_{i,2})$ to $\mathcal{A}$.

2. **Decryption Query.** For each query $\mathsf{ct}_j = (\mathsf{ct}_{j,1}, \mathsf{ct}_{j,2})$, the challenger first decrypts $\mathsf{ct}_{j,2}$ using secret key sk_2 as $\alpha_j||m'_j||r'_j = \mathsf{Dec}_2(\mathsf{sk}_2, \mathsf{ct}_{j,2})$. If $\alpha_j = 1$ or $\mathsf{ct}_1 \neq \mathsf{Enc}_1(\mathsf{sk}_1, m'_j; r'_j)$, it sends $\perp$ to $\mathcal{A}$. Else, it sends the message m'_j to $\mathcal{A}$.

- **Challenge Phase.** The adversary sends two challenge messages m_0^*, m_1^* to the challenger. The challenger chooses an n bit random string $r^* \leftarrow \{0,1\}^n$, and computes ciphertexts $\mathsf{ct}_1^*, \mathsf{ct}_2^*$ as $\mathsf{ct}_1^* = \mathsf{Enc}_1(\mathsf{sk}_1, m_b^*; r^*)$, $\mathsf{ct}_2 \leftarrow \mathsf{Enc}_2(\mathsf{sk}_2, 0||m_b^*||r^*)$, and sends the ciphertext $\mathsf{ct}^* = (\mathsf{ct}_1^*, \mathsf{ct}_2^*)$ to $\mathcal{A}$.
- **Post-Challenge Phase.** Identical to the pre-challenge phase, except the adversary is not allowed to query challenge ciphertext ct^* for decryption.
- **Guess.** The adversary sends its guess b' and wins if $b = b'$.

Game 1: This game is similar to previous game, except the challenger computes second component of each ciphertext (i.e., $\mathsf{ct}_{j,2}$ and ct_2^*) as an encryption of $1||0^{n+1}$. Also, each decryption query is answered differently if at least one of its components is same as that of any ciphertext that the challenger encrypted.

- **Setup Phase.** The challenger chooses secret key as $\mathsf{sk}_1 \leftarrow \mathsf{Setup}_1(1^\lambda)$, $\mathsf{sk}_2 \leftarrow \mathsf{Setup}_2(1^\lambda)$, bit $b \leftarrow \{0,1\}$. It sets $\mathsf{sk} = (\mathsf{sk}_1, \mathsf{sk}_2)$.
- **Pre-Challenge Phase.** The adversary is allowed to make following queries polynomially many times:
  1. **Encryption Query.** For each query m_i, the challenger chooses an n bit random string $r_i \leftarrow \{0,1\}^n$, and computes ciphertexts $\mathsf{ct}_{i,1}, \mathsf{ct}_{i,2}$ as $\mathsf{ct}_{i,1} = \mathsf{Enc}_1(\mathsf{sk}_1, m_i; r_i)$, $\mathsf{ct}_2 \leftarrow \mathsf{Enc}_2(\mathsf{sk}_2, 1||0^{n+1})$, and sends the ciphertext $\mathsf{ct}_i = (\mathsf{ct}_{i,1}, \mathsf{ct}_{i,2})$ to $\mathcal{A}$.
  2. **Decryption Query.** For each query $\mathsf{ct}_j = (\mathsf{ct}_{j,1}, \mathsf{ct}_{j,2})$, the challenger first checks if $\mathsf{ct}_j = \mathsf{ct}_i$ where ct_i is ciphertext corresponding to i^{th} message query. If $\mathsf{ct}_j = \mathsf{ct}_i$ for some i, then challenger sends m_i to $\mathcal{A}$. Else, if $\mathsf{ct}_{j,2} = \mathsf{ct}_{i,2}$ for some i, then it sends $\perp$ to $\mathcal{A}$. Else, it decrypts the ciphertext decrypts $\mathsf{ct}_{j,2}$ using secret key sk_2 as $\alpha_j||m'_j||r'_j = \mathsf{Dec}_2(\mathsf{sk}_2, \mathsf{ct}_{j,2})$. If $\alpha_j = 1$ or $\mathsf{ct}_{j,1} \neq \mathsf{Enc}_1(\mathsf{sk}_1, m'_j; r'_j)$, it sends $\perp$ to $\mathcal{A}$. Else, it sends the message m'_j to $\mathcal{A}$.
- **Challenge Phase.** The adversary sends two challenge messages m_0^*, m_1^* to the challenger. The challenger chooses a random string $r^* \leftarrow \{0,1\}^n$. It computes ciphertexts ct_1^* and ct_2^* as $\mathsf{ct}_1^* = \mathsf{Enc}_1(\mathsf{sk}_1, m_b^*; r^*)$, and $\mathsf{ct}_2^* \leftarrow \mathsf{Enc}_2(\mathsf{sk}_2, 1||0^{n+1})$. It sends $\mathsf{ct}^* = (\mathsf{ct}_1^*, \mathsf{ct}_2^*)$ to $\mathcal{A}$.
- **Post-Challenge Phase.** Identical to the pre-challenge phase, except the adversary is not allowed to query challenge ciphertext ct^* for decryption.
- **Guess.** The adversary sends its guess b' and wins if $b = b'$.

We now establish that no PPT adversary can distinguish between Game 0 and 1 with non-negligible advantage. To conclude, we also show that the advantage of any PPT adversary in the Game 1 is also negligible. Thus, the advantage of any PPT adversary in the Game 0 is also negligible.

Lemma A.1. If Π_2 is an IND-CCA secure encryption scheme, then for any PPT adversary $\mathcal{A}$, $|\mathsf{Adv}^0_{\mathcal{A}} - \mathsf{Adv}^1_{\mathcal{A}}| \leq \mathrm{negl}(\lambda)$ for some negligible function $\mathrm{negl}(\cdot)$.

Proof. For proving indistinguishability of Game 1 and 2, we need to sketch q intermediate hybrid games between these two, where q is the number of encryption queries made by $\mathcal{A}$(including the challenge query).[9] Observe that in Game 1, ciphertexts $\mathsf{ct}_{i,2}$ are encryptions of messages $(0||m_i||r_i)$[10]; however, in Game 2, ciphertexts $\mathsf{ct}_{i,2}$ are encryptions of $(1||0^{n+1})$. The high-level proof idea is to switch $\mathsf{ct}_{i,2}$ from encryptions

[9] In the description, we treat the challenge query as a standard encryption query. Concretely, if q_1 queries are made in pre-challenge phase, then challenge query is treated as the $(q_1+1)^{th}$ query i.e. $\mathsf{ct}_{q_1+1} = \mathsf{ct}^*$

[10] For challenge query, $m_i = m_b^*$

of $(0||m_i||r_i)$ to encryptions of $(1||0^{n+1})$ one-at-a-time by using IND-CCA security of Π_2. Concretely, k^{th} intermediate hybrid between Game 1 and 2 proceeds same as Game 1 except that the first k ciphertexts $\mathsf{ct}_{i,2}$ are computed as $\mathsf{ct}_{i,2} \leftarrow \mathsf{Enc}(\mathsf{sk}_2, 1||0^{n+1})$ i.e. for $i \leq k$, $\mathsf{ct}_{i,2}$ are encryptions of $(1||0^{n+1})$, and for $i > k$, $\mathsf{ct}_{i,2}$ are encryptions of $(0||m_i||r_i)$. For the analysis, Game 1 is regarded as 0^{th} intermediate hybrid, and Game 2 is regarded as q^{th} intermediate hybrid. Below we show that $\mathcal{A}$'s advantage in distinguishing any pair of consecutive intermediate hybrid is negligibly small.

We describe a reduction algorithm $\mathcal{B}$ which breaks the IND-CCA security of Π_2 if $\mathcal{A}$ distinguishes between hybrid games $k-1$ and k with non-negligible probability. $\mathcal{B}$ runs the Setup Phase (step 1) as in Game 0, except it does not choose secret key $\mathsf{sk}_2 \leftarrow \mathsf{Setup}_2(1^\lambda)$. For answering each message query (pre-challenge, challenge and post-challenge) m_i, it proceeds as follows. If $i < k$, it forwards $1||0^{n+1}$ to IND-CCA challenger, sets the challenger's response as $\mathsf{ct}_{i,2}$, encrypts $\mathsf{ct}_{i,1}$ as in Game 0 and sends $(\mathsf{ct}_{i,1}, \mathsf{ct}_{i,2})$ to $\mathcal{A}$. If $i = k$, it sends $(m_k, 1||0^{n+1})$ as its challenge messages, sets $\mathsf{ct}_{k,2}$ as the challenger's response, encrypts $\mathsf{ct}_{k,1}$ as in Game 0 and sends $(\mathsf{ct}_{k,1}, \mathsf{ct}_{k,2})$ to $\mathcal{A}$. Else if $i > k$, it forwards m_i to IND-CCA challenger, sets the challenger's response as $\mathsf{ct}_{i,2}$, encrypts $\mathsf{ct}_{i,1}$ as in Game 0 and sends $(\mathsf{ct}_{i,1}, \mathsf{ct}_{i,2})$ to $\mathcal{A}$. For answering each decryption query (pre-challenge and post-challenge) ct_j, it proceeds exactly as in Game 1. That means the challenger first checks if $\mathsf{ct}_j = \mathsf{ct}_i$ where ct_i is ciphertext corresponding to i^{th} message query. If $\mathsf{ct}_j = \mathsf{ct}_i$ for some i, then challenger sends m_i to $\mathcal{A}$. Else, if $\mathsf{ct}_{j,2} = \mathsf{ct}_{i,2}$ for some i, then it sends $\perp$ to $\mathcal{A}$. Else, it sends the ciphertext $\mathsf{ct}_{j,2}$ to IND-CCA challenger for decryption and sets the challenger's response as $\alpha_j||m'_j||r'_j$. If $\alpha_j = 1$ or $\mathsf{ct}_{j,1} \neq \mathsf{Enc}_1(\mathsf{sk}_1, m'_j; r'_j)$, it sends $\perp$ to $\mathcal{A}$. Else, it sends the message m'_j to $\mathcal{A}$. Finally, $\mathcal{A}$ outputs b', and $\mathcal{B}$ sends its guess as 0 (i.e. $(0||m^*_b||r^*)$ is encrypted) if $b = b'$, else it sends 1 as its guess to the IND-CCA challenger.

Note that the reduction algorithm $\mathcal{B}$ slightly deviates from honestly simulating the decryption queries in Game 0 as it always outputs $\perp$ if the second component of the queried ciphertext is same as $\mathsf{ct}_{i,2}$ for some i, or if the queried ciphertext is same as ct_i for some i. However, all such decryption queries will produce the same output (*i.e.* $\perp$ or m_i) during an honest simulation as well. This is because in Game 0 if any queried ciphertext ct_j is such that $\mathsf{ct}_j = \mathsf{ct}_i$, then the decryption of ct_i will be m_i as it was honestly encrypted. Similarly, if any queried ciphertext ct_j is such that $\mathsf{ct}_{j,2} = \mathsf{ct}_{i,2}$ and $\mathsf{ct}_{j,1} \neq \mathsf{ct}_{i,1}$ for some i, then decryption of ct_j shall be $\perp$ as the consistency check (i.e. $\mathsf{ct}_{j,1} \stackrel{?}{=} \mathsf{Enc}_1(\mathsf{pk}_1, m_i; r_i)$) will fail. Therefore, when the IND-CCA challenger encrypts $(0||m_k||r_k)$, then $\mathcal{B}$ exactly simulates the view of $(k-1)^{th}$ hybrid for $\mathcal{A}$. Otherwise if it encrypts $(1||0^{n+1})$, then $\mathcal{B}$ exactly simulates the view of k^{th} hybrid. Hence, if $|\mathsf{Adv}^0_{\mathcal{A}} - \mathsf{Adv}^1_{\mathcal{A}}|$ is non-negligible, then Π_2 is not an IND-CCA secure encryption scheme. ∎

Lemma A.2. If Π_1 is an IND-CPA secure encryption scheme, then for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^1_{\mathcal{A}} \leq \mathrm{negl}(\lambda)$ for some negligible function $\mathrm{negl}(\cdot)$.

Proof. The proof of this lemma follows directly from the IND-CPA security of Π_1. Consider a reduction algorithm $\mathcal{B}$ which plays the IND-CPA security game with encryption scheme Π_1 challenger. $\mathcal{B}$ runs the Setup Phase (step 1) as in Game 1, except it does not choose secret key $\mathsf{sk}_1 \leftarrow \mathsf{Setup}_1(1^\lambda)$. For answering each pre-challenge message query, $\mathcal{B}$ simply passes it to the IND-CPA challenger, receives the challenger's response $\mathsf{ct}_{i,1}$, encrypts $\mathsf{ct}_{i,2}$ as in Game 1 and sends $(\mathsf{ct}_{i,1}, \mathsf{ct}_{i,2})$ to $\mathcal{A}$. For answering each pre-challenge decryption query, $\mathcal{B}$ honestly simulates the decryption by first decrypting $\mathsf{ct}_{j,2}$ using the knowledge of sk_2, and then performing the consistency check using the decrypted value. Recall that $\mathcal{B}$ does not need the knowledge of sk_1 to perform the consistency check or perform decryption. Next, $\mathcal{A}$ sends two challenge messages m^*_0, m^*_1 to $\mathcal{B}$. $\mathcal{B}$ forwards these to the IND-CPA challenger as its challenge, and sets ct^*_1 as the corresponding challenge ciphertext. It computes ciphertext $\mathsf{ct}^*_2 \leftarrow \mathsf{Enc}_2(\mathsf{pk}_2, 1||0^{n+1})$, and sends $\mathsf{ct}^* = (\mathsf{ct}^*_1, \mathsf{ct}^*_2)$ as the challenge ciphertext to $\mathcal{A}$. For answering post-challenge queries, $\mathcal{B}$ proceeds same as in pre-challenge phase. Finally, $\mathcal{A}$ outputs b', and $\mathcal{B}$ sends the same bit b' as its guess to the IND-CPA challenger.

Note that $\mathcal{B}$ exactly simulates the view of Game 1 for $\mathcal{A}$as when IND-CPA challenger encrypts m^*_b, then ciphertext ct^* is also an encryption of m^*_b for $\mathcal{A}$. Therefore, if $\mathsf{Adv}^1_{\mathcal{A}}$ is non-negligible, then Π_1 is not an IND-CPA secure encryption scheme. ∎

■