Circular Security Separations for Arbitrary Length Cycles from LWE

Venkata Koppula
kvenkata@cs.utexas.edu

Brent Waters
bwaters@cs.utexas.edu*

Abstract

We describe a public key encryption that is IND-CPA secure under the Learning with Errors (LWE) assumption, but that is not circular secure for arbitrary length cycles. Previous separation results for cycle length greater than 2 require the use of indistinguishability obfuscation, which is not currently realizable under standard assumptions.

1 Introduction

The notion of key dependent message security departs from standard encryption security in that it allows the attacker to access ciphertexts where the messages are functions of the secret key. One prototypical example is k-circular security. An encryption scheme is said to be k-circular secure, if an adversary is unable to distinguish $\mathsf{Enc}(\mathsf{pk}_1, \mathsf{sk}_k), \mathsf{Enc}(\mathsf{pk}_2, \mathsf{sk}_1), \ldots, \mathsf{Enc}(\mathsf{pk}_k, \mathsf{sk}_{k-1})$ from k encryptions of the all 0 message.

The demand for encryption schemes that provide circular security has arisen in multiple applications. Camenisch and Lysyanskaya [14] applied circular secure encryption to anonymous credential systems, while Laud [22] and Adão et al.[2] use circular security to prove the soundness of symbolic protocols. Most notably Gentry's [20] bootstrapping technique shows how to achieve fully homomorphic encryption (FHE) for circuits of any depth chosen at evaluation time (i.e. not fixed at setup) from those of shallower depth if the FHE scheme is circular secure. There have been multiple constructions of circular secure schemes or more generally key-dependent message security, some proven in the random oracle model [14, 9] and others in the standard model from particular assumptions [10, 6, 11, 7, 12, 5, 4].

One interesting question is whether k-circular security can come "for free". Is there some k such that *any* IND-CPA secure encryption scheme is guaranteed to be k-circular secure? If true, this would give an immediate path to applying Gentry's FHE bootstrapping technique among other applications.

A trivial folklore argument provides a separation for the case of $k = 1$. The first non-trivial example for $k = 2$ was given by Acar et al. [1] and extended by Cash, Green and Hohenberger [15] using the Decisional Diffie-Hellman assumption over asymmetric bilinear groups. Subsequently, Bishop, Hohenberger and Waters [8] extended the result to include symmetric groups under the decision linear assumption as well as moving to the lattice setting with a separation under the Learning with Errors (LWE) assumption. However, they leave open the possibility of getting "free" circular security by simply extending the key cycle lengths to be greater than two.

The more general case of k-length cycles for arbitrary size k was considered by Koppula, Ramchen, and Waters [21] who showed that under the assumption of indistinguishability obfuscation (for polynomial sized circuits), for any k there exists schemes that are IND-CPA secure, but that are not k-circular secure. Marcedone and Orlandi [23] independently gave a similar result, but under the assumption of a virtual black box secure obfuscator for a certain functionality.

*Supported by NSF CNS-0952692, CNS-1228599 and CNS-1414082. DARPA through the U.S. Office of Naval Research under Contract N00014-11-1-0382, Google Faculty Research award, the Alfred P. Sloan Fellowship, Microsoft Faculty Fellowship, and Packard Foundation Fellowship.

While these works cast doubts on the ability to get free circular security for larger cycle lengths, they do so by invoking a quite strong primitive of obfuscation. Notably, the only current candidates for obfuscation rely on the multilinear encodings for which the first candidate was proposed in 2013[18]. In addition, to being relatively untested there have subsequently been several attacks discovered [16, 17] on various multilinear encoding proposals.

Separation without obfuscation. This brings up to the central question of this paper.

Can we separate IND-CPA and circular security for arbitrary length cycles using standard assumptions (i.e without invoking obfuscation or multilinear maps)?

Such a result would provide a firmer understanding of circular security. In addition, the introduction of the first general purpose obfuscation candidate [19] has lead to the realization of many cryptographic primitives that to this point were not realizable (e.g., deniable encryption, functional encryption, etc.). However, very few of these newly realized primitives have since been adapted to a standard assumption — one not involving obfuscation or multilinear maps. We believe that attacking this problem for one primitive can begin to crack the ice and hopefully begin to lead to insights for others.

A separation example from Learning with Errors The main result of our paper is the introduction of a family of encryption systems that are IND-CPA secure under the LWE assumption, but which are made to not be k-circular secure for arbitrary k. Here, k can be any polynomial in the security parameter.

We first illustrate the challenges of building such a scheme by looking at the recent Bishop, Hohenberger and Waters construction [8], which gave a separation from LWE for $k = 2$. In this work, Bishop et al. first proposed a general framework for constructing circular security separations. This framework, called the *k-cycle tester framework*, consists of algorithms for setup, key generation, encryption and testing cycles. Note that unlike an encryption scheme, there is no decryption algorithm here. The setup algorithm outputs public parameters, which are used by the key generation algorithm to choose the public key and secret key. The encryption algorithm takes a public key and a message, and outputs its encryption. The cycle tester algorithm takes k public keys and k encryptions, and outputs 1 if the k public keys/ciphertexts form a key cycle (that is, $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \mathsf{sk}_{i-1})$), else it outputs 0 with all but negligible probability. For security, encryptions of distinct messages must be computationally indistinguishable. Bishop et al. showed how to use such a k-cycle tester, together with an IND-CPA encryption scheme, to construct an IND-CPA encryption scheme that is not k-circular secure. They also showed several constructions of a 2-cycle tester from various assumptions, including one from LWE.

The BHW 2-cycle tester from LWE: Unlike most existing LWE based encryption schemes where the message is part of a large norm vector, Bishop et al. used a novel approach for encrypting the message: via 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 used to compute, for any matrix $\mathbf{U}$, a low norm matrix $\mathbf{S} = \mathbf{A}^{-1}(\mathbf{U})$ such that $\mathbf{A} \cdot \mathbf{S} = \mathbf{U}$. [1] Moreover, if $\mathbf{U}$ is chosen uniformly at random, then $\mathbf{S}$ reveals no information about the matrix $\mathbf{A}$, or the randomness used to sample $\mathbf{A}$, $T_{\mathbf{A}}$. Bishop et al. used the message vector as randomness for the lattice trapdoor generation algorithm.

Their construction (with some modifications) can be described as follows. The setup algorithm simply outputs the LWE parameters. The key generation algorithm first samples a matrix $\mathbf{A}$ along with its lattice trapdoor $T_{\mathbf{A}}$. The secret key is the randomness used to compute $\mathbf{A}, T_{\mathbf{A}}$. To compute the public key, the algorithm chooses a matrix $\mathbf{C}$, computes $\mathbf{D} = \mathbf{C} \cdot \mathbf{A} + \mathsf{noise}$ and outputs $(\mathbf{C}, \mathbf{D})$ as the public key. The encryption algorithm uses the message msg as randomness for the trapdoor generation algorithm, computing a matrix $\mathbf{Z}$ and its trapdoor $T_{\mathbf{Z}}$. Next, it chooses a uniformly random $\{-1, 1\}$ vector $\mathbf{r}$ and computes $\mathbf{u} = \mathbf{C}^\top \cdot \mathbf{r}$ and $\mathbf{v} = \mathbf{D}^\top \cdot \mathbf{r} \approx \mathbf{A}^\top \cdot \mathbf{C}^\top \cdot \mathbf{r}$. The final ciphertext consists of a short vector $\mathbf{s} = \mathbf{Z}^{-1}(\mathbf{u})$ that

[1] 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 SamplePre.

contains the message, and a large vector $\mathbf{v}$ that is used for cycle testing. For $\mathsf{IND\text{-}CPA}$ security, one can use the LWE assumption and the Leftover Hash Lemma to argue that $\mathbf{C}^\top \cdot \mathbf{r}$ is indistinguishable from a uniformly random vector, and therefore $\mathbf{Z}^{-1}(\mathbf{C}^\top \cdot \mathbf{r})$ reveals no information about msg.

The cycle testing algorithm takes as input two ciphertexts $(\mathbf{v}_1, \mathbf{s}_1)$, $(\mathbf{v}_2, \mathbf{s}_2)$ and checks if $\mathbf{v}_1^\top \cdot \mathbf{s}_2$ is close to $\mathbf{v}_2^\top \cdot \mathbf{s}_1$. To see why this works, let us consider the case when the two ciphertexts form a key cycle; that is, $\mathbf{s}_1 = \mathbf{B}_2^{-1}(\mathbf{C}_1^\top \cdot \mathbf{r}_1)$, $\mathbf{v}_1^\top = \mathbf{r}_1^\top \cdot \mathbf{C}_1 \cdot \mathbf{B}_1 + \mathsf{noise}$ and $\mathbf{s}_2 = \mathbf{B}_1^{-1}(\mathbf{C}_2^\top \cdot \mathbf{r}_2)$, $\mathbf{v}_2^\top = \mathbf{r}_2^\top \cdot \mathbf{C}_2 \cdot \mathbf{B}_2 + \mathsf{noise}$. In this case, the testing algorithm outputs 1 because $\mathbf{v}_1^\top \cdot \mathbf{s}_2 \approx \mathbf{r}_1^\top \cdot \mathbf{C}_1 \cdot \mathbf{C}_2^\top \cdot \mathbf{r}_2 = \mathbf{r}_2^\top \cdot \mathbf{C}_2 \cdot \mathbf{C}_1^\top \cdot \mathbf{r}_1 \approx \mathbf{v}_2^\top \cdot \mathbf{s}_1$. However, if both ciphertexts are encryptions of $\mathbf{0}$, then both $\mathbf{v}_1^\top \cdot \mathbf{s}_2$ and $\mathbf{v}_2^\top \cdot \mathbf{s}_1$ are uniformly random elements, and therefore, they are likely not close to each other. At a high level, this approach works because in a key cycle, the $\mathbf{B}_1$ in $\mathbf{v}_1$ and $\mathbf{B}_1^{-1}$ in $\mathbf{s}_2$ cancel each other (and similarly the matrix $\mathbf{B}_2$ and $\mathbf{B}_2^{-1}$ in $\mathbf{v}_2$ and $\mathbf{s}_1$ respectively).

Unfortunately, the BHW approach cannot be directly used to handle longer cycles.

Our approach via cascading cancellations: For simplicity, let us consider the problem of constructing a 3-cycle tester (this can be easily extended to handle longer cycles). The starting point of our approach is the following simple observation: for $i = 1, 2, 3$, let $\mathbf{B}_i$ be matrices with trapdoors, and let $\mathbf{X}, \mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3$ be arbitrary matrices. Consider the matrices $\mathbf{M}_1 = \mathbf{B}_3^{-1}(\mathbf{C}_1 \cdot \mathbf{X})$, $\mathbf{M}_2 = \mathbf{B}_1^{-1}(\mathbf{C}_2 \cdot \mathbf{B}_2)$ and $\mathbf{M}_3 = \mathbf{B}_2^{-1}(\mathbf{C}_3 \cdot \mathbf{B}_3)$. Then $\mathbf{B}_1 \cdot \mathbf{M}_2 \cdot \mathbf{M}_3 \cdot \mathbf{M}_1 = \mathbf{C}_2 \cdot \mathbf{C}_3 \cdot \mathbf{C}_1 \cdot \mathbf{X}$. The matrix $\mathbf{B}_1$ starts the 'chain reaction' by canceling $\mathbf{B}_1^{-1}$ in $\mathbf{M}_1$, and after each matrix multiplication, the product is a canceling matrix for the next one in the sequence.

In fact, this observation can be easily extended to have noisy matrices: for $i = 1, 2, 3$, let $\mathbf{B}_i$ be matrices with trapdoors, $\mathbf{C}_i$ matrices with low norm entries, and $\mathbf{X}$ any arbitrary matrix. Consider the matrices $\mathbf{M}_1 = \mathbf{B}_3^{-1}(\mathbf{C}_1 \cdot \mathbf{X} + \mathsf{noise})$, $\mathbf{M}_2 = \mathbf{B}_1^{-1}(\mathbf{C}_2 \cdot \mathbf{B}_2 + \mathsf{noise})$ and $\mathbf{M}_3 = \mathbf{B}_2^{-1}(\mathbf{C}_3 \cdot \mathbf{B}_3 + \mathsf{noise})$. Then $\mathbf{B}_1 \cdot \mathbf{M}_2 \cdot \mathbf{M}_3 \cdot \mathbf{M}_1 \approx \mathbf{C}_2 \cdot \mathbf{C}_3 \cdot \mathbf{C}_1 \cdot \mathbf{X}$. This observation inspires us to try the following approach: each ciphertext consists of two low norm matrices such that a key cycle gives us two parallel chains with the same end product matrix. Before discussing this approach in more detail, we will present an extension of the cycle tester framework which will help simplify our presentation.

Extending the BHW k-cycle tester framework: We introduce an extension of the BHW cycle tester framework, which we call the *Leader-Follower k-cycle tester* framework. This framework has a setup algorithm for outputting the parameters, two different key generation and encryption algorithms, and finally a tester algorithm. Looking ahead, in our counterexample, one of the public keys/ciphertexts has a special role, and they are generated using the 'leader' key generation/encryption algorithms, while the remaining are generated using the 'follower' key generation/encryption algorithms. For correctness, we require that the test algorithm outputs 1 if the k ciphertexts form an encryption cycle, else it outputs 0. For security, both the leader and follower encryption schemes must satisfy $\mathsf{IND\text{-}CPA}$ security. One can establish a simple reduction from our Leader-Follower framework to the BHW cycle-tester framework.

First Attempt via two parallel chains: As an initial attempt, we present a Leader-Follower 3-cycle tester where any message/secret key consists of two strings, each of which can be used to sample a lattice trapdoor. To begin, we will describe the follower key generation/encryption algorithms.

The follower key generation algorithm chooses two strings $\mathbf{x}_1, \mathbf{x}_2$ and sets $(\mathbf{x}_1, \mathbf{x}_2)$ as the secret key. To compute the public key, it first chooses two matrices $\mathbf{B}_1, \mathbf{B}_2$ with trapdoors (using strings $\mathbf{x}_1$ and $\mathbf{x}_2$ respectively as randomness). The public key simply consists of the matrices $\mathbf{B}_1, \mathbf{B}_2$. The corresponding encryption algorithm uses the message $\mathsf{msg} = (\mathbf{y}_1, \mathbf{y}_2)$ to sample matrices $\mathbf{Z}_1, \mathbf{Z}_2$ together with the respective trapdoors. Next, it chooses a low norm matrix $\mathbf{C}$ and outputs $\mathbf{S}_1 = \mathbf{Z}_1^{-1}(\mathbf{C} \cdot \mathbf{B}_1 + \mathsf{noise})$, $\mathbf{S}_2 = \mathbf{Z}_2^{-1}(\mathbf{C} \cdot \mathbf{B}_2 + \mathsf{noise})$ as the ciphertext.

The leader key generation algorithm is a bit more involved. The secret key is chosen as in the follower key generation, and the public key has an additional component: a uniformly random matrix $\mathbf{X}$. As in the follower encryption algorithm, the leader encryption algorithm chooses matrices $\mathbf{Z}_1, \mathbf{Z}_2$ and their trapdoors. Next, it chooses low norm matrices $\mathbf{C}_1$ and outputs $\mathbf{S}_1 = \mathbf{Z}_1^{-1}(\mathbf{C} \cdot \mathbf{X} + \mathsf{noise})$, $\mathbf{S}_2 = \mathbf{Z}_2^{-1}(\mathbf{C} \cdot \mathbf{X} + \mathsf{noise})$.

The testing algorithm, on input three ciphertexts $(\mathbf{S}_{11}, \mathbf{S}_{12})$, $(\mathbf{S}_{21}, \mathbf{S}_{22})$, $(\mathbf{S}_{31}, \mathbf{S}_{32})$ and three public keys

$(\mathbf{B}_{11}, \mathbf{B}_{12}, \mathbf{X})$, $(\mathbf{B}_{21}, \mathbf{B}_{22})$, $(\mathbf{B}_{31}, \mathbf{B}_{32})$, checks if $\mathbf{B}_{11} \cdot \mathbf{S}_{21} \cdot \mathbf{S}_{31} \cdot \mathbf{S}_{11} \approx \mathbf{B}_{12} \cdot \mathbf{S}_{22} \cdot \mathbf{S}_{32} \cdot \mathbf{S}_{12}$. The testing algorithm works as desired, because if $\mathbf{C}_i$ is the random matrix used for computing $\mathbf{S}_{ij}$ and the three ciphertexts form an encryption cycle, then both the expressions are approximately $\mathbf{C}_2 \cdot \mathbf{C}_3 \cdot \mathbf{C}_1 \cdot \mathbf{X}$.

For $\mathsf{IND\text{-}CPA}$ security of follower key generation/encryptions, note that by the LWE assumption, both $\mathbf{C} \cdot \mathbf{B}_1 + \mathsf{noise}$ and $\mathbf{C} \cdot \mathbf{B}_2 + \mathsf{noise}$ are indistinguishable from truly random matrices. As a result, $\mathbf{S}_1$ and $\mathbf{S}_2$ hide the randomness used to choose $\mathbf{Z}_1$ and $\mathbf{Z}_2$.

Next, let us consider $\mathsf{IND\text{-}CPA}$ security of leader key generation/encryptions. Unfortunately, this part is problematic, because the matrices $\mathbf{Z}_1^{-1}(\mathbf{C} \cdot \mathbf{X} + \mathsf{noise})$ and $\mathbf{Z}_2^{-1}(\mathbf{C} \cdot \mathbf{X} + \mathsf{noise})$ clearly reveal information about $\mathbf{Z}_1$ and $\mathbf{Z}_2$ (for example, one can check if $\mathbf{Z}_1 = \mathbf{Z}_2$). To address this problem, we first increase the number of parallel chains to a suitably large number (say ℓ), and have ℓ matrices $\mathbf{X}_1, \ldots, \mathbf{X}_\ell$ as part of the leader public key. These matrices satisfy the following relation: there exist a $\{-1, 1\}$ coefficient vector $\mathbf{x}$ such that $x_i \cdot \mathbf{X}_i = 0$. This vector $\mathbf{x}$ must be hidden from the $\mathsf{IND\text{-}CPA}$ adversary; however, the test algorithm must be able to somehow use this vector to cancel out the $\mathbf{X}_i$ matrices.

Our solution: Our final solution is similar to the approach outlined above. The messages and secret keys consist of ℓ strings, each of which can be used as the randomness for lattice trapdoor generation. The follower key generation and encryption algorithms are similar to the ones described above, except that now there are ℓ public matrices $\mathbf{B}_1, \ldots, \mathbf{B}_\ell$, and the ciphertext consists of ℓ low norm matrices $\mathbf{S}_1, \ldots, \mathbf{S}_\ell$, where $\mathbf{S}_i = \mathbf{Z}_i^{-1}(\mathbf{C} \cdot \mathbf{B}_i + \mathsf{noise})$.

The main differences are in the leader key generation algorithm. The leader key generation algorithm first uses the secret key to sample ℓ matrices $\mathbf{B}_1, \ldots, \mathbf{B}_\ell$ along with their trapdoors. Next, it chooses an ℓ length string $\mathbf{x}$, chooses $\ell - 1$ matrices $\mathbf{X}_1, \ldots, \mathbf{X}_{\ell-1}$, and sets X_ℓ such that $\sum_i x_i \cdot \mathbf{X}_i = 0$. The public key consists of the matrices $(x_1 \cdot \mathbf{B}_1, \ldots, x_\ell \cdot \mathbf{B}_\ell, \mathbf{X}_1, \ldots, \mathbf{X}_\ell)$ (note the x_i coefficients attached to each $\mathbf{B}_i$). To encrypt a message, one chooses matrices $\mathbf{Z}_1, \ldots, \mathbf{Z}_\ell$ with trapdoors using the message strings as randomness. Then it chooses a matrix $\mathbf{C}$ and outputs $\mathbf{S}_i = \mathbf{Z}_i^{-1}(\mathbf{C} \cdot \mathbf{X}_i + \mathsf{noise})$ as the ciphertext. To argue $\mathsf{IND\text{-}CPA}$ security, note that the matrices $\mathbf{X}_i$ look uniformly random (since the vector $\mathbf{x}$ is hidden from the adversary). As a result, the matrices $\mathbf{C} \cdot \mathbf{X}_i + \mathsf{noise}$ look like ℓ uniformly random matrices, and therefore the adversary does not learn any information about the $\mathbf{Z}_i$ matrices.

The test algorithm is similar to what was described in the previous solution, except at the end, the algorithm computes the sum of the final products, and checks if it is of low norm. The correctness of the test algorithm for $k = 3$ can be verified easily from the table below. Let $\mathsf{pk}_1 = (x_1 \cdot \mathbf{B}_{11}, \ldots, x_\ell \cdot \mathbf{B}_{i\ell}, \mathbf{X}_1, \ldots, \mathbf{X}_\ell)$, and let $\mathsf{ct}_1, \mathsf{ct}_2, \mathsf{ct}_3$ be the three ciphertexts that form a 3-cycle.

pk_1 matrices	$x_1 \cdot \mathbf{B}_{11}$	$\ldots$	$x_\ell \cdot \mathbf{B}_{1\ell}$
ct_2 matrices	$\mathbf{S}_{21} = \mathbf{B}_{11}^{-1}(\mathbf{C}_2 \cdot \mathbf{B}_{21} + \mathsf{noise})$	$\ldots$	$\mathbf{S}_{2\ell} = \mathbf{B}_{1\ell}^{-1}(\mathbf{C}_2 \cdot \mathbf{B}_{2\ell} + \mathsf{noise})$
ct_3 matrices	$\mathbf{S}_{31} = \mathbf{B}_{21}^{-1}(\mathbf{C}_3 \cdot \mathbf{B}_{31} + \mathsf{noise})$	$\ldots$	$\mathbf{S}_{3\ell} = \mathbf{B}_{2\ell}^{-1}(\mathbf{C}_3 \cdot \mathbf{B}_{3\ell} + \mathsf{noise})$
ct_1 matrices	$\mathbf{S}_{11} = \mathbf{B}_{31}^{-1}(\mathbf{C}_1 \cdot \mathbf{X}_1 + \mathsf{noise})$	$\ldots$	$\mathbf{S}_{1\ell} = \mathbf{B}_{3\ell}^{-1}(\mathbf{C}_1 \cdot \mathbf{X}_\ell + \mathsf{noise})$
Product	$\approx x_1 \cdot \mathbf{C}_2 \cdot \mathbf{C}_3 \cdot \mathbf{C}_1 \cdot \mathbf{X}_1$	$\ldots$	$\approx x_\ell \cdot \mathbf{C}_2 \cdot \mathbf{C}_3 \cdot \mathbf{C}_1 \cdot \mathbf{X}_\ell$

Clearly, the sum of the products is low norm, and therefore the testing algorithm outputs 1 if the input is a key cycle. For a non-key cycle, each of these products is a uniformly random matrix, and therefore, with high probability, their sum has large norm. This concludes our scheme.

The construction described above can be extended to handle any polynomially bounded cycle length k. For this, we need to set the matrix dimensions n, m, the LWE noise parameter σ and the LWE modulus q appropriately as functions of k and security parameter λ.

Recent Work Independent of and concurrently to our work, Alamati and Peikert [3] provided novel constructions for separating circular security and $\mathsf{IND\text{-}CPA}$ security. While the goals of [3] and our work are similar, the technical approaches of these two works, interestingly, differ quite significantly.

Alamati and Peikert present a tester in the common random string model whose IND-CPA security is based on the Ring-LWE assumption, which reduces to worst case problems in ideal lattices. While their testing algorithm also employs a "telescoping cancellation", the other technical aspects differ quite significantly from ours. Intuitively, their tester works by transforming a key cycle into two ciphertexts that are encrypted under similar randomness.[2] This transformation relies upon the commutativity property of rings. In addition, their underlying encryption system is designed such that it is detectable if there exists two ciphertexts encrypted under similar randomness. Putting together the transformation and the common randomness detection properties results in a cycle detector.

Their ring-based solution has the property that the encryption system grows polynomially in the security parameter and the cycle length detection k. Since the overhead increases only polynomially with k, it is possible to create an encryption system where the (maximum) length of cycles that can be tested is a polynomial $k(\lambda)$ that grows with the security parameter λ.

In their second solution they turn toward adapting their techniques with the goal of achieving security under the plain LWE assumption. The primary barrier is in moving outside the ring setting they no longer have the commutativity property needed to transform key cycles into two ciphertexts under similar randomness. Instead they manage to realize a form commutativity by employing a novel tensoring technique while maintaining standard LWE security. The cost, however, of this technique is an overhead factor of n^k where n is the lattice security parameter. Since the system's overhead increases exponentially in k, the maximum cycle length for any system must be a constant and cannot grow polynomially in λ like the ring-based solution.

Our system is based on the plain LWE assumption and has polynomial growth in k. Thus, it is possible for to build a system where the maximum cycle length $k(\lambda)$ is an arbitrary polynomial. In addition, if one fixes the LWE parameters n, m and modulus q to be fixed functions of the security parameter we can do without a common random string (assuming all parties use the same security parameter). For simplicity, we do not highlight these features in our main presentation, however, we address them further at an informal level in Section 4.1.

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 $\mathcal{D}$.

Given a randomized algorithm $A(\cdot)$, the notation $A(\cdot;\cdot)$ is used to explicitly describe the randomness used by A (e.g. $A(x;r)$ denotes computation on input x using randomness r).

Randomness Extraction: We will use the following theorem, which follows from the Leftover Hash Lemma.

Theorem 2.1. Let $m > (n+1)\log_2 q + \omega(\log n)$ and q a prime. Then the statistical distance between the following distributions is negligible in n.

$$\{(\mathbf{A}, \mathbf{A}\cdot\mathbf{r}) : \mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}, \mathbf{r} \leftarrow \{-1,1\}^m\} \approx \{(\mathbf{A}, \mathbf{u}) : \mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}, \mathbf{u} \leftarrow \mathbb{Z}_q^m\}.$$

Public Key Encryption A public key encryption scheme $\mathcal{PKE}$ with message space $\mathcal{M}$ consists of algorithms Setup, KeyGen, Enc, Dec with the following syntax.

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$. The setup algorithm takes as input the security parameter and outputs the public parameters pp.
- $\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{pk}, \mathsf{sk})$. The key generation algorithm takes as input the public parameters pp and outputs a public key pk and secret key sk.

[2]Here "similar randomness" means ignoring short noise factors.

- $\mathsf{Enc}(\mathsf{pk}, m \in \mathcal{M}) \to \mathsf{ct}$. The encryption algorithm takes as input a public key pk and a message $m \in \mathcal{M}$. It outputs a ciphertext ct.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \to y \in \mathcal{M} \cup \{\perp\}$. The decryption algorithm takes as input a secret key sk, ciphertext ct and outputs a message $y \in \mathcal{M}$ if the decryption is successful, else it outputs $\perp$.

A public key encryption scheme must satisfy correctness and IND-CPA security.

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

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

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

Security: In this work, we will be using the IND-CPA security notion.

Definition 2.1. Let $\mathcal{PKE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a public key encryption scheme. The scheme is said to be IND-CPA secure if for all security parameters λ, all PPT adversaries $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathcal{PKE},\lambda} = |\Pr[\mathcal{A}$ wins the IND-CPA game $] - 1/2|$ is negligible in λ, where the IND-CPA experiment is defined below:

- The challenger chooses $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ and sends pk to $\mathcal{A}$.
- The adversary sends two challenge messages m_0, m_1 to the challenger. The challenger chooses $b \leftarrow \{0, 1\}$ and sends $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$ to $\mathcal{A}$.
- $\mathcal{A}$ sends its guess b' and wins if $b = b'$.

2.1 Lattice Preliminaries

An m-dimensional lattice $\mathcal{L}$ is a discrete additive subgroup of $\mathbb{R}^m$. Given positive integers n, m, q and a matrix $A \in \mathbb{Z}_q^{n \times m}$, we let $\Lambda_q^\perp(A)$ denote the lattice $\{x \in \mathbb{Z}^m \ : \ Ax = 0 \mod q\}$. For $u \in \mathbb{Z}_q^n$, we let $\Lambda_q^u(A)$ denote the set $\{x \in \mathbb{Z}^m \ : \ Ax = u \mod q\}$.

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 $\mathcal{L} \subset \mathbb{R}^m$, 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 [25]) 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 $n \times m$, and $\mathcal{L} = \Lambda_q^\perp(\mathbf{A})$. Then

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

Learning with Errors (LWE) The Learning with Errors (LWE) problem was introduced by Regev [28]. 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:

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

Under a quantum reduction, Regev [28] 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 [27].

Theorem 2.2 ([28])**.** 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 [26, 13] 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 [3]:

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

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 2.2. 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 $\ell_{\mathrm{TG}}(n)$ bits of randomness, 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 $\sigma \in \mathbb{R}$ (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 any string $\mathbf{y} \in \{0,1\}^{\ell_{\mathrm{TG}}}$, vector $\mathbf{u}$ and parameter σ, let $(\mathbf{A}, T_\mathbf{A}) \leftarrow \mathsf{TrapGen}(1^n, 1^m; \mathbf{y})$, $\mathbf{s} \leftarrow \mathsf{SamplePre}(\mathbf{A}, T_\mathbf{A}, \mathbf{u}, \sigma)$. Then $\mathbf{A} \cdot \mathbf{s} = \mathbf{u}$.
2. *Well Distributedness of Matrix:* The following distributions are statistically indistinguishable:
$$\{\mathbf{A} : (\mathbf{A}, T_\mathbf{A}) \leftarrow \mathsf{TrapGen}(1^n, 1^m)\} \approx_s \{\mathbf{A} : \mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}\}.$$
3. *Well Distributedness of Preimage:* For any string $\mathbf{y} \in \{0,1\}^{\ell_{\mathrm{TG}}}$, let $(\mathbf{A}, T_\mathbf{A}) = \mathsf{TrapGen}(1^n, 1^m; \mathbf{y})$. Then if $\sigma = \omega(\sqrt{n \cdot \log q \cdot \log m})$, the following distributions are statistically indistinguishable:
$$\{\mathbf{s} : \mathbf{u} \leftarrow \mathbb{Z}_q^n, \mathbf{s} \leftarrow \mathsf{SamplePre}(\mathbf{A}, T_\mathbf{A}, \mathbf{u}, \sigma)\} \approx_s \mathcal{D}_{\mathbb{Z}^m, \sigma}.$$

Note that the first and third properties must be satisfied for all strings $y \in \{0,1\}^{\ell_{\mathrm{TG}}}$. These properties are satisfied by the gadget-based trapdoor lattice sampler of [24].

[3] 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.

3 Circular Security and Our Framework for Generating Circular Counterexamples

In this section, we define the notion of circular security for public key encryption schemes, and then discuss frameworks for obtaining separation between circular security and $\mathsf{IND\text{-}CPA}$ security. Let $\mathcal{PKE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a public key encryption scheme. A k-encryption cycle consists of k encryptions, where the i^{th} encryption is an encryption of the $(i-1)^{th}$ secret key using the i^{th} public key. Intuitively, the scheme is k-circular secure if no adversary can distinguish between an encryption cycle and k encryptions of zeros.

Definition 3.1 (k-Circular Security)**.** Let $\mathcal{PKE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a public key cryptosystem. The scheme is said to k-circular secure if for all PPT adversaries $\mathcal{A}$, the following expression is at most $\text{negl}(\lambda)$.

$$\left| \Pr\left[1 \leftarrow \mathcal{A}(\{(\mathsf{pk}_i, \mathsf{ct}_i)\}_i) \ : \ \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda); (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp}); \\ \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \mathsf{sk}_{i-1}) \end{array} \right] \right.$$
$$\left. - \Pr\left[1 \leftarrow \mathcal{A}(\{(\mathsf{pk}_i, \mathsf{ct}_i)\}_i) \ : \ \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda); (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp}); \\ \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, 0^{|\mathsf{sk}_{i-1}|}) \end{array} \right] \right|.$$

The above definition is derived from the Key-Dependent Message (KDM) security notion of Black et al. [9]. A weaker security notion, proposed by Cash et al. [15] requires the adversary to output the secret key when given an encryption cycle. Koppula et al. [21] showed that if there exists an adversary that can distinguish between an encryption cycle and encryptions of zeros, then there exists an adversary that can recover the entire secret key given an encryption cycle. Therefore, in this work, we focus on Definition 3.1.

3.1 The BHW Cycle Tester Framework

In a recent work, Bishop, Hohenberger and Waters [8] introduced a generic framework for creating circular security counterexamples. In this *cycle tester* framework, there are four algorithms - Setup, KeyGen, Encrypt and *Test*. The setup algorithm outputs the public parameters, the key generation algorithm uses the public parameters to output a public key/secret key pair. The encryption algorithm takes a public key and message as input, and outputs a ciphertext. Finally, the testing algorithm takes as input k public keys and k ciphertexts, and outputs 1 if the k encryptions form an encryption cycle, else it outputs 0. *Note that in this framework, there is no decryption algorithm.* The security requirement is identical to the $\mathsf{IND\text{-}CPA}$ security game. The following description is taken from [8].

Definition 3.2 (k-Cycle Tester)**.** A *cycle tester* $\Gamma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Test})$ for message space $\mathcal{M}$ and secret key space $\mathcal{S}$ is a tuple of algorithms specified as follows:

- $\mathsf{Setup}(1^\lambda, 1^k) \rightarrow \mathsf{pp}$. The setup algorithm takes as input the security parameter λ and the length of cycle k. It outputs the public parameters pp.
- $\mathsf{KeyGen}(\mathsf{pp}) \rightarrow (\mathsf{pk}, \mathsf{sk})$. The key generation algorithm takes as input the public parameters pp and outputs a public key pk and secret key $\mathsf{sk} \in S$.
- $\mathsf{Enc}(\mathsf{pk}, m \in M) \rightarrow C$. The encryption algorithm takes as input a public key pk and a message $m \in M$ and outputs a ciphertext C.
- $\mathsf{Test}(\mathbf{pk}, \mathbf{ct}) \rightarrow \{0, 1\}$. On input $\mathbf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_k)$ and $\mathbf{ct} = (\mathsf{ct}_1, \ldots, \mathsf{ct}_k)$, the testing algorithm 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 (given by the following expression) is at least $1/p(\lambda)$.

$$\Pr\left[1 \leftarrow \mathsf{Test}(\mathbf{pk}, \mathbf{ct}) \ : \ \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda); (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp}); \\ \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{sk}_{i-1}) \end{array}\right]$$
$$- \Pr\left[1 \leftarrow \mathsf{Test}(\mathbf{pk}, \mathbf{ct}) \ : \ \begin{array}{c} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda); (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(1^\lambda); \\ \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}_i, 0^{|\mathsf{sk}_{i-1}|}) \end{array}\right]$$

2. (IND-CPA Security) Let $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \cdot)$ be an encryption scheme with empty decryption algorithm. The scheme Π must satisfy the IND-CPA security definition (see Definition 2.1).

Bishop et al. [8] showed that in order to construct a separation between IND-CPA and k-circular security, it suffices to construct a k-cycle tester (as defined in Definition 3.2).

Theorem 3.1 (CPA Counterexample from Cycle Testers, [8])**.** If there exists an IND-CPA-secure encryption scheme Π for message space $M = (M_1 \times M_2)$ and secret key space $S_1 \subseteq M_1$ and an k-cycle tester Γ for message space M_2 and secret key space $S_2 \subseteq M_2$, then there exists an IND-CPA-secure encryption scheme Π' for message space $M = (M_1 \times M_2)$ and secret key space $S = (S_1 \times S_2)$ that is k-circular insecure.

3.2 Our Leader-Follower Tester Framework

In this section, we propose an adaptation of the BHW cycle tester framework that we call *Leader-Follower Tester*. In this modification, the key generation and encryption have two modes - leader and follower. The tester algorithm takes k public keys and ciphertexts: the first public key (resp. ciphertext) is a 'leader' public key (resp. ciphertext). The remaining are 'follower' public keys/ciphertexts. It outputs 1 if the ciphertexts form a cycle, else it outputs 0. First, we will formally define the syntax/properties of this modification, and then show how this implies the cycle tester framework of [8].

Definition 3.3 (k-Leader-Follower Tester)**.** A *Leader-Follower cycle tester* $\Gamma = $ (Setup, KeyGen-L, KeyGen-F, Enc-L, Enc-F, Test) for message space M and secret key space S is a tuple of algorithms specified as follows:

- $\mathsf{Setup}(1^\lambda, 1^k) \rightarrow \mathsf{pp}$. The setup algorithm takes as input the security parameter n and length of cycle k, and outputs public parameters pp.
- $\mathsf{KeyGen\text{-}L}(\mathsf{pp}) \rightarrow (\mathsf{pk}, \mathsf{sk})$. The leader key generation algorithm takes as input the public parameters pp, and outputs a public key pk and secret key $\mathsf{sk} \in S$.
- $\mathsf{KeyGen\text{-}F}(\mathsf{pp}) \rightarrow (\mathsf{pk}, \mathsf{sk})$. The follower key generation algorithm takes as input the public parameters pp, and outputs a public key pk and secret key $\mathsf{sk} \in S$.
- $\mathsf{Enc\text{-}L}(\mathsf{pk}, m \in M) \rightarrow C$. The leader encryption algorithm takes as input a leader public key pk and a message $m \in M$ and outputs a ciphertext C.
- $\mathsf{Enc\text{-}F}(\mathsf{pk}, m \in M) \rightarrow C$. The follower encryption algorithm takes as input a follower public key pk and a message $m \in M$ and outputs a ciphertext C.
- $\mathsf{Test}(\mathbf{pk}, \mathbf{ct}) \rightarrow \{0, 1\}$. The test algorithm takes as input a public key vector $\mathbf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_k)$ and a ciphertext vector $\mathbf{ct} = (\mathsf{ct}_1, \ldots, \mathsf{ct}_k)$. Of these, the first public key and ciphertext are of leader type, while the remaining are of follower type. The testing algorithm 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 (given by the following expression) is at least $1/p(\lambda)$.

$$\Pr\left[1 \leftarrow \mathsf{Test}(\{\mathsf{pk}_i, \mathsf{ct}_i\}) \;:\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^k); (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{KeyGen\text{-}L}(\mathsf{pp}); \\ \mathsf{ct}_1 \leftarrow \mathsf{Enc\text{-}L}(\mathsf{pk}_1, \mathsf{sk}_k); (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen\text{-}F}(\mathsf{pp}); \\ \mathsf{ct}_i \leftarrow \mathsf{Enc\text{-}F}(\mathsf{pk}, \mathsf{sk}_{i-1}) \end{array}\right]$$
$$- \Pr\left[1 \leftarrow \mathsf{Test}(\{\mathsf{pk}_i, \mathsf{ct}_i\}) \;:\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^k); (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{KeyGen\text{-}L}(\mathsf{pp}); \\ \mathsf{ct}_1 \leftarrow \mathsf{Enc\text{-}L}(\mathsf{pk}_1, \mathsf{sk}_k); (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen\text{-}F}(\mathsf{pp}); \\ \mathsf{ct}_i \leftarrow \mathsf{Enc\text{-}F}(\mathsf{pk}_i, 0^{|\mathsf{sk}_{i-1}|}) \end{array}\right]$$

2. (IND-CPA Security for Both Modes) Let $\Pi\text{-}\mathsf{L} = (\mathsf{Setup}, \mathsf{KeyGen\text{-}LEnc\text{-}L}, \cdot)$ and $\Pi\text{-}\mathsf{F} = (\mathsf{Setup}, \mathsf{KeyGen\text{-}F}, \mathsf{Enc\text{-}F}, \cdot)$ be two encryption schemes with empty decryption algorithm. We require that both Π-L and Π-F must satisfy IND-CPA security as in Definition 2.1.

We will now show that the Leader-Follower Tester defined above implies the tester framework of [8] (Definition 3.2).

Lemma 3.1. Suppose there exists a k-Leader-Follower-Tester (Setup, KeyGen-L, Enc-L, KeyGen-F, Enc-F, Test) as defined in Definition 3.3. Then there exists a k-Tester (Setup$'$, KeyGen$'$, Enc$'$, Test$'$) that satisfies Definition 3.2.

Proof. The proof of this lemma is fairly straightforward: the setup algorithm first chooses a bit $b \leftarrow \mathsf{Ber}_{1/k}$. If $b = 1$, it runs KeyGen-L and sets the mode to be 'leader', else it runs KeyGen-F and sets the mode to be 'follower'. The encryption algorithm, based on the mode, either uses Enc-L or Enc-F.

- $\mathsf{Setup}'(1^\lambda) \rightarrow (\mathsf{pp})$: The setup algorithm first chooses the public parameters pp.
- $\mathsf{KeyGen}'(\mathsf{pp}) \rightarrow (\mathsf{pk}', \mathsf{sk}')$: The key generation algorithm chooses $b \leftarrow \mathsf{Ber}_{1/k}$. If $b = 1$, it chooses $(\mathsf{pk}_L, \mathsf{sk}_L) \leftarrow \mathsf{KeyGen\text{-}L}(1^\lambda)$, sets $\mathsf{mode} = L$ and $\mathsf{pk}' = (\mathsf{pk}_L, \mathsf{mode})$, $\mathsf{sk}' = \mathsf{sk}_L$. Else, it chooses $(\mathsf{pk}_F, \mathsf{sk}_F) \leftarrow \mathsf{KeyGen\text{-}F}(1^\lambda)$, sets $\mathsf{mode} = F$ and $\mathsf{pk}' = (\mathsf{pk}_F, \mathsf{mode})$.
- $\mathsf{Enc}'(\mathsf{pk}', \mathsf{msg}) \rightarrow \mathsf{ct}$: The encryption algorithm parses the public key pk' as $(\mathsf{pk}, \mathsf{mode})$. If $\mathsf{mode} = L$, it computes $\mathsf{ct} \leftarrow \mathsf{Enc\text{-}L}(\mathsf{pk}, m)$, else it computes $\mathsf{ct} \leftarrow \mathsf{Enc\text{-}F}(\mathsf{pk}, m)$.
- $\mathsf{Test}'((\mathsf{pk}_1, \ldots, \mathsf{pk}_k), (\mathsf{ct}_1, \ldots, \mathsf{ct}_k)) \rightarrow \{0, 1\}$: The test algorithm first checks that pk_1 has mode 'L' and all other public keys have mode 'F'. If not, it aborts and outputs 1 with probability 1/2. Else, it simply runs $\mathsf{Test}((\mathsf{pk}_1, \ldots, \mathsf{pk}_k), (\mathsf{ct}_1, \ldots, \mathsf{ct}_k))$ and outputs the result.

Now, given the scheme $(\mathsf{Setup}', \mathsf{KeyGen}', \mathsf{Enc}', \mathsf{Dec}')$, we need to show that it satisfies Definition 3.2.

- *Testing Correctness*: Suppose the algorithm Test succeeds with non-negligible advantage ϵ. From the construction, it follows that the algorithm Test succeeds with probability $(1 - 1/k)^{k-1} \cdot (1/k) \cdot \epsilon$. This is because if pk_1 has mode 'L' and all others have mode 'F', then the algorithm has advantage ϵ, else it has advantage 0.
- *Security*: The IND-CPA security of $(\mathsf{Setup}', \mathsf{Enc}', \mathsf{Test}')$ follows directly from the Leader/Follower IND-CPA security definitions (recall that the leader-follower-tester scheme must satisfy IND-CPA security in both modes).

■

The purpose of introducing this modification is that it simplifies the description of our construction (see Section 4). In our construction, one of the k public keys/ciphertexts is used to 'tie' the ends together, and therefore is referred to as the Leader. A similar structure can be found in the counterexample shown by [21].

4 Counter Example for k-Circular Security

In this section, we will describe a Leader-Follower cycle tester $\mathcal{E} = (\mathsf{Setup}, \mathsf{KeyGen\text{-}L}, \mathsf{Enc\text{-}L}, \mathsf{KeyGen\text{-}F}, \mathsf{Enc\text{-}F}, \mathsf{Test})$ such that it satisfies the properties described in Definition 3.3. Recall, $\ell_{\mathrm{TG}}(n)$ denote the number of bits of randomness required by the $\mathsf{TrapGen}$ algorithm. For simplicity of description, we will drop the dependence on n.

Fix any $\epsilon < 1/2$ and cycle length k. Our scheme has following algorithms:

- $\mathsf{Setup}(1^\lambda)$: The setup algorithm chooses the following parameters: matrix dimensions n, m, LWE modulus q, parameter σ for the Gaussian noise distribution χ, and an additional parameter ℓ. These parameters will be functions of λ, k and ϵ. We require the parameters to satisfy the following relations:

  $n = \mathsf{poly(k, \lambda)}$

  (n, m) are the dimensions of matrices output by $\mathsf{TrapGen}$, therefore $m = \Omega(n \cdot \log q)$.

  $\chi = \mathcal{D}_\sigma$ and $\sigma/q \geq \mathsf{poly(n)}/2^{\mathsf{n}^\epsilon}$ (for LWE noise/modulus ratio to be greater than $\mathsf{poly(n)}/2^{\mathsf{n}^\epsilon}$)

  $\ell = \Omega(n \cdot \log q)$ (for Leftover Hash Lemma: Lemma 2.1)

  $\ell \cdot (\ell \cdot m \cdot n \cdot \sigma)^k \leq q/8$(for the correctness of our Test algorithm)

  One instantiation which works is as follows: let $n = k^{1/\epsilon} \cdot \lambda$, $m = 2n \cdot \log q$, $\sigma = n^c$ for some constant c. Then setting $q = 2^{n^\epsilon}$, $\ell = 2n \log q$ satisfies the above relations.

  Note that if k is constant, we can set q to be polynomial (which will in turn result in polynomial approximation factors for GapSVP).

  The message space of our scheme (which is also the space of secret keys) is $(\{0, 1\}^{\ell_{\mathrm{TG}}})^\ell$.

- $\mathsf{KeyGen\text{-}L}(\mathsf{pp})$: The leader key generation algorithm first chooses $\mathbf{y}_1, \ldots \mathbf{y}_w \leftarrow \{0, 1\}^{\ell_{\mathrm{TG}}}$. For $i \leq w$, the algorithm computes $(\mathbf{B}_i, T_{\mathbf{B}_i}) = \mathsf{TrapGen}(1^n; \mathbf{y}_i)$. Next it chooses a string $\mathbf{x} \in \{-1, 1\}^\ell$ by choosing uniformly random bits $x_i \leftarrow \{-1, 1\}$ for $i \leq \ell - 1$ and setting $x_\ell = 1$. The first part of the public key consists of matrices $\mathbf{D}_i$ defined as follows:

  $$\mathbf{D}_i = x_i \cdot \mathbf{B}_i \in \mathbb{Z}_q^{n \times m} \quad \text{for all } i \leq \ell$$

  Next, it selects random vectors $\mathbf{h}_i \in \mathbb{Z}_q^n$ for $i < \ell$ and lets $\mathbf{h}_\ell = -\sum_{i<\ell} x_i \cdot \mathbf{h}_i$. The second part of the public key consists of the vectors $\{\mathbf{h}_i\}_i$.

  The secret key is $\mathsf{sk} = \{\mathbf{y}_i\}_{i \leq \ell}$ and the public key is $\mathsf{pk} = (\{\mathbf{D}_i\}_{i \leq \ell}, \{\mathbf{h}_i\}_{i \leq \ell})$.

- $\mathsf{Enc\text{-}L}(\mathsf{pk}, \mathsf{msg})$:

  Let $\mathsf{pk} = \{\mathbf{D}_i\}, \{\mathbf{h}_i\}$ and $\mathsf{msg} = (\mathbf{m}_1, \ldots, \mathbf{m}_\ell)$. The leader encryption algorithm computes $(\mathbf{Z}_i, T_{\mathbf{Z}_i}) = \mathsf{TrapGen}(1^n; \mathbf{m}_i)$ for $i \leq \ell$. Next, it chooses matrix $\mathbf{C} \leftarrow \chi^{n \times n}$, error vector $\mathbf{e}_i \leftarrow \chi^n$ for $i \leq \ell$, and sets $\mathbf{f}_i = \mathbf{C} \cdot \mathbf{h}_i + \mathbf{e}_i$. Finally, it computes $\mathbf{s}_i \leftarrow \mathsf{SamplePre}(\mathbf{Z}_i, T_{\mathbf{Z}_i}, \sigma, \mathbf{f}_i)$.

  The ciphertext is set to be $\mathsf{ct} = (\mathbf{s}_1, \ldots, \mathbf{s}_\ell)$.

- $\mathsf{KeyGen\text{-}F}(\mathsf{pp})$: The follower setup algorithm takes as input the security parameter 1^n. It first chooses ℓ uniformly random binary vectors of length ℓ_{TG}; that is, it chooses $\mathbf{y}_i \leftarrow \{0, 1\}^{\ell_{\mathrm{TG}}}$ for $i \leq \ell$. Next, it computes $(\mathbf{B}_i, T_{\mathbf{B}_i}) = \mathsf{TrapGen}(1^n; \mathbf{y}_i)$.

  The algorithm outputs secret key $\mathsf{sk} = \{\mathbf{y}_i\}_{i \leq \ell}$ and public key $\mathsf{pk} = \{\mathbf{B}_i\}_{i \leq \ell}$.

- $\mathsf{Enc\text{-}F}(\mathsf{pk}, \mathsf{msg})$: Let $\mathsf{msg} = (\mathbf{m}_1, \ldots, \mathbf{m}_\ell)$. The follower encryption algorithm computes $(\mathbf{Z}_i, T_{\mathbf{Z}_i}) = \mathsf{TrapGen}(1^n; \mathbf{m}_i)$ for $i \leq \ell$. Next, it chooses matrix $\mathbf{C} \leftarrow \chi^{n \times n}$, error matrix $\mathbf{E}_i \leftarrow \chi^{n \times m}$ and sets $\mathbf{F}_i = \mathbf{C} \cdot \mathbf{B}_i + \mathbf{E}_i$. Finally, it computes $\mathbf{S}_i \leftarrow \mathsf{SamplePre}(\mathbf{Z}_i, T_{\mathbf{Z}_i}, \sigma, \mathbf{F}_i)$.

  The ciphertext is set to be $\mathsf{ct} = (\mathbf{S}_1, \ldots, \mathbf{S}_\ell)$.

- $\mathsf{Test}((\mathsf{pk}^{(1)}, \ldots, \mathsf{pk}^{(k)}), (\mathsf{ct}^{(1)}, \ldots, \mathsf{ct}^{(k)}))$ **:**

  Let $\mathsf{pk}^{(1)} = (\{\mathbf{D}_i^{(1)}\}, \{\mathbf{h}_i\})$, $\mathsf{ct}^{(1)} = (\mathbf{s}_1^{(1)}, \ldots, \mathbf{s}_\ell^{(1)})$ and $\mathsf{ct}^{(j)} = (\mathbf{S}_1^{(j)}, \ldots, \mathbf{S}_\ell^{(j)})$ for $2 \leq j \leq k$.

  The test algorithm computes
  $$\mathsf{sum} = \sum_{i \in [\ell]} \mathbf{D}_i^{(1)} \cdot (\prod_{2 \leq j \leq k} \mathbf{S}_i^{(j)}) \cdot \mathbf{s}_i^{(1)}.$$

  It tests if each component of $\mathsf{sum} \in [-q/8, q/8]$ and outputs 1 if so to indicate a cycle. Otherwise it outputs 0.

4.1 Discussion

Before proving the correctness and security properties of our construction we give a brief interlude to highlight various features including the ability to remove setup and letting the maximum cycle length be a polynomial function of the security parameter.

Testing for polynomial length cycles : The construction described above works for any (apriori fixed) polynomial bound $k = k(\lambda)$ on the length of the cycle. This is achieved by setting the parameters n, q, σ appropriately. For large k, we require σ/q to be subexponential in n. However, if k is a constant, then we can have q to be polynomial in n. In particular, we can set $n = \lambda$, $q = n^{5k}$, $\sigma = n$ and $m = 2n \log q$. This setting of parameters results in polynomial approximation factors for worst-case lattice problems like GapSVP.

Removing Setup : For simplicity of presentation, we have a separate setup algorithm which chooses the LWE parameters. Since this algorithm can be deterministic once k, ϵ is fixed, we don't need a setup algorithm. This can be performed by the key generation algorithm itself.

Handling cycles of length less than k : Our tester algorithm description assumes the length of the cycle is k. However, note that the same algorithm can be used to handle cycle lengths less than k as well. Let $(\mathsf{pk}^{(1)}, \ldots, \mathsf{pk}^{(t)})$ be t public keys, and let $(\mathsf{ct}^{(1)}, \ldots, \mathsf{ct}^{(t)})$ be t ciphertexts, where $t \leq k$. Further, let us assume $\mathsf{pk}^{(1)} = (\{\mathbf{D}_i^{(1)}\}, \{\mathbf{h}_i\})$ and $\mathsf{ct}^{(1)} = (\mathbf{s}_1^{(1)}, \ldots, \mathbf{s}_\ell^{(1)})$ and $\mathsf{ct}^{(j)} = (\mathbf{S}_1^{(j)}, \ldots, \mathbf{S}_\ell^{(j)})$ for $2 \leq j \leq t$. The test algorithm simply computes $\sum_{i \in \ell} \mathbf{D}_i^{(1)} \cdot (\prod_{2 \leq j \leq t} \mathbf{S}_i^{(j)}) \cdot \mathbf{s}_i^{(1)}$ and checks if each component of the sum is less than $q/8$.

We would like to mention that in general, any cycle-tester where the test algorithm works for only specific cycle lengths can be transformed to one where the test algorithm works for all lengths below a specific length bound. More formally, suppose for all integers k, there exists a cycle tester $\mathcal{T}_k$ where the test algorithm can distinguish between a k-cycle and k encryptions of $\mathbf{0}$. Then we can construct a cycle tester $\mathcal{T}'_k$ that can detect cycles of all lengths less than k. This scheme $\mathcal{T}'_k$ will simply consist of k separate instantiations of $\mathcal{T}_i$, one for each cycle length $i \leq k$.

Improving efficiency of our scheme : In our scheme, the public keys consist of ℓ matrices, each message msg in the message space consists of ℓ vectors $(\mathbf{m}_1, \ldots, \mathbf{m}_\ell)$ and the ciphertext for msg comprises of ℓ different matrices, where the i^{th} one is derived from $\mathbf{m}_i$. We require $\ell = \Omega(n \log q)$ since we use an information-theoretic argument for the final step of our proof. Alternatively, we can reduce ℓ to $O(n)$ by choosing the vector $\mathbf{x}$ in $\mathsf{KeyGen\text{-}L}$ from $\mathcal{D}_\sigma$, and using the LWE assumption instead of the Leftover Hash Lemma.

4.2 Proof of Correctness

First, we will show that the Test algorithm distinguishes between a cycle and encryptions of zeros with overwhelming probability. For this, we need to set up some notations. Let $\mathsf{Bd} = n \cdot \sigma$. From Lemma 2.1, it follows that if $\mathbf{x} \leftarrow \chi^n$, then $\|\mathbf{x}\|_\infty \leq \mathsf{Bd}$ with overwhelming probability. Let $\mathsf{pk}^{(1)} = (\{\mathbf{D}_i\}, \{\mathbf{h}_i\})$ where $\mathbf{D}_i = x_i \cdot \mathbf{B}_i^{(1)}$. Recall, the vectors $\mathbf{h}_i$ are chosen such that $\sum_i x_i \cdot \mathbf{h}_i = 0$ and therefore, $\|\mathbf{h}_i\|_\infty \leq \ell \cdot \mathsf{Bd}$.

Next, the follower public keys are $\mathsf{pk}^{(p)} = \{\mathbf{B}_i^p\}$ for $2 \leq p \leq k$ and $T_{\mathbf{B}_i^p}$ denote the trapdoor corresponding to matrix $\mathbf{B}_i^p$ for $p \leq k, i \leq \ell$.

We will first analyse the case where the ciphertexts are encryptions of a cycle. Let $\mathsf{ct}^{(1)} = (\mathbf{s}_1, \ldots, \mathbf{s}_\ell)$. Here, $\mathbf{f}_i = \mathbf{C}^{(1)} \cdot \mathbf{h}_i + \mathbf{e}_i$ and $\mathbf{s}_i = \mathsf{SamplePre}(\mathbf{B}_i^{(k)}, T_{\mathbf{B}_i^{(k)}}, \sigma, \mathbf{f}_i)$.

Next, for $2 \leq p \leq k$, let $\mathbf{F}_i^{(p)} = \mathbf{C}^{(p)} \cdot \mathbf{B}_i^{(p)} + \mathbf{E}_i^{(p)}$ and $\mathbf{S}_i^{(p)} = \mathsf{SamplePre}(\mathbf{B}_i^{(p-1)}, T_{\mathbf{B}_i^{(p-1)}}, \sigma, \mathbf{F}_i^{(p)})$. Let $\Delta_{i,p^*} = \mathbf{D}_i \cdot (\prod_{p=2}^{p^*} \mathbf{S}_i^{(p)})$ and $\Delta'_{i,p^*} = x_i \cdot (\prod_{p=2}^{p^*} \mathbf{C}^{(p)}) \cdot \mathbf{B}_i^{(p^*)}$

Claim 4.1. For any $i \leq \ell, p^* \in [2, k]$,

$$\|\Delta_{i,p^*} - \Delta'_{i,p^*}\|_\infty \leq (\ell \cdot m \cdot \mathsf{Bd})^{p^*-1}.$$

Proof. The proof of this theorem involves a simple induction argument on p^*. First, the base case: $p^* = 2$. In this case, $\Delta_{i,p^*} = \mathbf{D}_i \cdot \mathbf{S}_i^{(2)} = x_i \cdot \mathbf{C}^{(2)} \cdot \mathbf{B}_i^{(2)} + x_i \cdot \mathbf{E}_i^{(2)}$. Hence $\|\Delta_{i,2} - \Delta'_{i,2}\|_\infty \leq \ell \cdot m \cdot \mathsf{Bd}$.

Suppose this holds true for all indices less than p^*. Now, $\Delta_{i,p^*} = \Delta_{i,p^*-1} \cdot \mathbf{S}_i^{(p^*)}$, and let $\Delta_{i,p^*-1} = \Delta'_{i,p^*-1} + \mathsf{Err}_{i,p^*-1}$, where $\|\mathsf{Err}_{i,p^*-1}\|_\infty \leq (\ell \cdot m \cdot \mathsf{Bd})^{p^*-2}$.

$$\begin{aligned}
\Delta_{i,p^*} &= \Delta'_{i,p^*-1} \cdot \mathbf{S}_i^{(p^*)} + \mathsf{Err}_{i,p^*-1} \cdot \mathbf{S}_i^{(p^*)} \\
&= \Delta'_{i,p^*} + x_i \cdot (\prod_{p=2}^{p^*-1} \mathbf{C}^{(p)}) \cdot \mathbf{E}_i^{(p^*)} + \mathsf{Err}_{i,p^*-1} \cdot \mathbf{S}_i^{(p^*)}
\end{aligned}$$

Let $\mathsf{Err}_{i,p^*} = x_i \cdot (\prod_{p=2}^{p^*-1} \mathbf{C}^{(p)}) \cdot \mathbf{E}_i^{(p^*)} + \mathsf{Err}_{p^*-1} \cdot \mathbf{S}_i^{(p^*)}$.

$$\begin{aligned}
\|\mathsf{Err}_{i,p^*}\|_\infty &\leq (\ell \cdot n \cdot \mathsf{Bd})^{p^*-2} \cdot (\ell \cdot m \cdot \mathsf{Bd}) + \|\mathsf{Err}_{p^*-1}\|_\infty \cdot (m \cdot \mathsf{Bd}) \\
&\leq (\ell \cdot n \cdot \mathsf{Bd})^{p^*-2} \cdot (\ell \cdot m \cdot \mathsf{Bd}) + (\ell \cdot m \cdot \mathsf{Bd})^{p^*-2} \cdot (m \cdot \mathsf{Bd}) \\
&\leq (\ell \cdot m \cdot \mathsf{Bd})^{p^*-1}
\end{aligned}$$

■

Finally, let us now consider the term $\Delta_k \cdot \mathbf{s}_i$. By a similar analysis as above, we can show that $\Delta_k \cdot \mathbf{s}_i = x_i \cdot (\prod_{p=2}^{k} \mathbf{C}^{(p)}) \cdot \mathbf{C}^{(1)} \cdot \mathbf{h}_i + \mathsf{Error_i}$ where $\|\mathsf{Error_i}\|_\infty \leq (\ell \cdot m \cdot \mathsf{Bd})^k$. As a result,

$$\|\sum_i \Delta_k \cdot \mathbf{s}_i\|_\infty = \|\sum_i x_i \cdot \mathbf{h}_i\|_\infty + \sum_i \|\mathsf{Error_i}\|_\infty \leq \ell \cdot (\ell \cdot m \cdot \mathsf{Bd})^k.$$

Given our choice of parameters, $\ell \cdot (\ell \cdot m \cdot \mathsf{Bd})^k < q/8$, and as a result, the Test algorithm outputs 1.

On the other hand, if the k cycle consists of encryptions of $\mathbf{0}$, then for all $i \leq \ell$, $\mathbf{D}_i \cdot \prod_{p=2}^{k} \mathbf{S}_i^{(p)} \cdot \mathbf{s}_i$ is a uniformly random vector in $\mathbb{Z}_q^n$, and therefore the test algorithm outputs 1 with negligible probability.

4.3 Proof of INDCPA Security

In this section, we will show that the construction described above is $\mathsf{IND\text{-}CPA}$ secure as per Definition 3.3. Recall, the $\mathsf{IND\text{-}CPA}$ security definition for leader-based encryption schemes requires two separate $\mathsf{IND\text{-}CPA}$ proofs for both leader and follower modes.

4.3.1 INDCPA security for Leader Mode

First, we will prove $\mathsf{IND\text{-}CPA}$ security for Leader mode. For this, we will define a sequence of hybrid experiments, and then show that the hybrids are computationally indistinguishable. The first hybrid will correspond to the $\mathsf{IND\text{-}CPA}$ security game, while the final hybrid will be one where the adversary has 0 advantage.

Hyb_0**:** This corresponds to the $\mathsf{IND\text{-}CPA}$ security game.

1. Setup Phase:
   (a) The challenger first chooses $\mathbf{y}_i \leftarrow \{0,1\}^{\ell_{\mathrm{TG}}}$ for $i \leq \ell$ and computes $(\mathbf{B}_i, T_{\mathbf{B}_i}) = \mathsf{TrapGen}(1^n; \mathbf{y}_i)$.
   (b) Next, it chooses $x_i \leftarrow \{-1, 1\}$ for $i < \ell$, sets $x_\ell = 1$.
   (c) It chooses $\mathbf{h}_i \leftarrow \mathbb{Z}_q^n$ for $i < \ell$, sets $\mathbf{h}_\ell = -\sum_{i<\ell} x_i \cdot \mathbf{h}_i$.
   (d) Finally, the challenger sends $(\{x_i \cdot \mathbf{B}_i\}, \{\mathbf{h}_i\})$ to the adversary.

2. Challenge Phase
   (a) The adversary sends two messages $\mathsf{msg}_0, \mathsf{msg}_1$. The challenger chooses matrix $\mathbf{C} \leftarrow \chi^{n\times n}$, error vector $vece_i \leftarrow \chi^n$ for $i \leq \ell$ and sets $\mathbf{f}_i = \mathbf{C} \cdot \mathbf{h}_i + \mathbf{e}_i$ for $i \leq \ell$.
   (b) Next, it chooses $b \leftarrow \{0, 1\}$. Let $\mathsf{msg}_b = (\mathbf{m}_1, \ldots, \mathbf{m}_\ell)$. The challenger computes $(\mathbf{Z}_i, T_{\mathbf{Z}_i}) = \mathsf{TrapGen}(1^n; \mathbf{m}_i)$.
   (c) Using $T_{\mathbf{Z}_i}$, the challenger computes $\mathbf{s}_i \leftarrow \mathsf{SamplePre}(T_{\mathbf{Z}_i}, \mathbf{f}_i)$ for all $i \leq \ell$. It sends $\mathsf{ct}^* = (\{\mathbf{s}_i\})$.

3. Guess: The adversary sends its guess b' and wins if $b = b'$.

Hyb_1**:** In this game, the challenger chooses $\mathbf{B}_i$ uniformly at random, and outputs $\{\mathbf{B}_i\}$ as part of public key, instead of $\{x_i \cdot \mathbf{B}_i\}$.

1. Setup Phase:
   (a) The challenger first chooses $\mathbf{B}_i \leftarrow \mathbb{Z}_q^{n\times m}$.
   (b) Next, it chooses $x_i \leftarrow \{-1, 1\}$ for $i < \ell$, sets $x_\ell = 1$.
   (c) It chooses $\mathbf{h}_i \leftarrow \mathbb{Z}_q^n$ for $i < \ell$, sets $\mathbf{h}_\ell = -\sum_{i<\ell} x_i \cdot \mathbf{h}_i$.
   (d) Finally, the challenger sends $(\{\mathbf{B}_i\}, \{\mathbf{h}_i\})$ to the adversary.

2. Challenge Phase
   (a) The adversary sends two messages $\mathsf{msg}_0, \mathsf{msg}_1$. The challenger chooses matrix $\mathbf{C} \leftarrow \chi^{n\times n}$, error vector $vece_i \leftarrow \chi^n$ for $i \leq \ell$ and sets $\mathbf{f}_i = \mathbf{C} \cdot \mathbf{h}_i + \mathbf{e}_i$ for $i \leq \ell$.
   (b) Next, it chooses $b \leftarrow \{0, 1\}$. Let $\mathsf{msg}_b = (\mathbf{m}_1, \ldots, \mathbf{m}_\ell)$. The challenger computes $(\mathbf{Z}_i, T_{\mathbf{Z}_i}) = \mathsf{TrapGen}(1^n; \mathbf{m}_i)$.
   (c) Using $T_{\mathbf{Z}_i}$, the challenger computes $\mathbf{s}_i \leftarrow \mathsf{SamplePre}(\mathbf{Z}_i, T_{\mathbf{Z}_i}, \sigma, \mathbf{f}_i)$ for all $i \leq \ell$. It sends $\mathsf{ct}^* = (\{\mathbf{s}_i\})$.

3. Guess: The adversary sends its guess b' and wins if $b = b'$.

Hyb_2**:** In this game, the challenger chooses $\mathbf{h}_\ell$ uniformly at random instead of setting it as $-\sum x_i \mathbf{h}_i$. Therefore, from this game onwards, the challenger does not need to choose x_i for $i < \ell$.

1. Setup Phase:
   (a) The challenger first chooses $\mathbf{B}_i \leftarrow \mathbb{Z}_q^{n \times m}$.
   (b) It chooses $\mathbf{h}_i \leftarrow \mathbb{Z}_q^n$ for $i \leq \ell$.
   (c) Finally, the challenger sends $(\{\mathbf{B}_i\}, \{\mathbf{h}_i\})$ to the adversary.

2. Challenge Phase
   (a) The adversary sends two messages $\mathsf{msg}_0, \mathsf{msg}_1$. The challenger chooses matrix $\mathbf{C} \leftarrow \chi^{n \times n}$, error vector $vece_i \leftarrow \chi^n$ for $i \leq \ell$ and sets $\mathbf{f}_i = \mathbf{C} \cdot \mathbf{h}_i + \mathbf{e}_i$ for $i \leq \ell$.
   (b) Next, it chooses $b \leftarrow \{0, 1\}$. Let $\mathsf{msg}_b = (\mathbf{m}_1, \ldots, \mathbf{m}_\ell)$. The challenger computes $(\mathbf{Z}_i, T_{\mathbf{Z}_i}) = \mathsf{TrapGen}(1^n; \mathbf{m}_i)$.
   (c) Using $T_{\mathbf{Z}_i}$, the challenger computes $\mathbf{s}_i \leftarrow \mathsf{SamplePre}(\mathbf{Z}_i, T_{\mathbf{Z}_i}, \sigma, \mathbf{f}_i)$ for all $i \leq \ell$. It sends $\mathsf{ct}^* = (\{\mathbf{s}_i\})$.

3. Guess: The adversary sends its guess b' and wins if $b = b'$.

Hyb_3**:** In this game, the challenger modifies the challenge phase. It chooses uniformly random vectors $\mathbf{f}_i \leftarrow \mathbb{Z}_q^n$.

1. Setup Phase:
   (a) The challenger first chooses $\mathbf{B}_i \leftarrow \mathbb{Z}_q^{n \times m}$. $\mathbf{h}_i \leftarrow \mathbb{Z}_q^n$ for $i \leq \ell$ and sends $(\{\mathbf{B}_i\}, \{\mathbf{h}_i\})$ to the adversary.

2. Challenge Phase
   (a) The adversary sends two messages $\mathsf{msg}_0, \mathsf{msg}_1$.
   The challenger chooses $\mathbf{f}_i \leftarrow \mathbb{Z}_q^n$ for all $i \leq \ell$.
   (b) Next, it chooses $b \leftarrow \{0, 1\}$. Let $\mathsf{msg}_b = (\mathbf{m}_1, \ldots, \mathbf{m}_\ell)$. The challenger computes $(\mathbf{Z}_i, T_{\mathbf{Z}_i}) = \mathsf{TrapGen}(1^n; \mathbf{m}_i)$.
   (c) Using $T_{\mathbf{Z}_i}$, the challenger computes $\mathbf{s}_i \leftarrow \mathsf{SamplePre}(\mathbf{Z}_i, T_{\mathbf{Z}_i}, \sigma, \mathbf{f}_i)$ for all $i \leq \ell$. It sends $\mathsf{ct}^* = (\{\mathbf{s}_i\})$.

3. Guess: The adversary sends its guess b' and wins if $b = b'$.

Hyb_4**:** In this game, the challenger chooses $\mathbf{s}_i$ from the Discrete Gaussian distribution $\mathcal{D}_{\mathbb{Z}^m, \sigma}$ with parameter σ. Note that in this hybrid, the adversary has 0 advantage.

1. Setup Phase:
   (a) The challenger chooses $\mathbf{v}_{i,j} \leftarrow \mathbb{Z}_q^n$, for $i \leq \ell$, $j \leq l$ and sends $\mathsf{pk} = (\{\mathbf{B}_i\}, \{\mathbf{v}_{i,j})$ to the adversary.

2. Challenge Phase
   (a) The adversary sends two messages $\mathsf{msg}_0, \mathsf{msg}_1$.
   The challenger chooses $\mathbf{d}_i \leftarrow \mathbb{Z}_q^n$ for all $i \leq \ell$.

(b) Next, the challenger chooses bit $b \leftarrow \{0,1\}$ and $\mathbf{s}_i \leftarrow \mathcal{D}_\sigma$. It sends $\mathsf{ct}^* = (\{\mathbf{d}_i\}, \{\mathbf{s}_i\})$.

3. Guess: The adversary sends its guess b' and wins if $b = b'$.

Analysis: We will now show that any PPT adversary has nearly identical advantage in the hybrid experiments described above. Let $\mathsf{Adv}^i_{\mathcal{A}}$ denote the advantage of adversary $\mathcal{A}$ in experiment Hyb_i.

Claim 4.2. For any adversary $\mathcal{A}$, $\mathsf{Adv}^0_{\mathcal{A}} - \mathsf{Adv}^1_{\mathcal{A}} \leq \text{negl}(n)$.

Proof. We will show that the statistical distance between the distributions of public keys in Hyb_0 and Hyb_1 is negligible in the security parameter n. Note that the only difference between the two hybrids is the distribution of $\mathbf{B}_i$ for $i \leq \ell$.

From the well-distributedness property of $\mathsf{TrapGen}$, we know that the following distributions have negligible statistical distance:

$$\{\mathbf{B}_i \ : \ (\mathbf{B}_i, T_{\mathbf{B}_i}) \leftarrow \mathsf{TrapGen}(1^n)\} \approx \{\mathbf{B}_i \ : \ \mathbf{B}_i \leftarrow \mathbb{Z}_q^{n\times m}\}.$$

Next, note that the following distributions are identical:

$$\{(x_i, x_i \cdot \mathbf{B}_i) \ : \ x_i \leftarrow \{-1,1\}, \mathbf{B}_i \leftarrow \mathbb{Z}_q^{n\times m}\} \equiv \{(x_i, \mathbf{B}_i) \ : \ x_i \leftarrow \{-1,1\}, \mathbf{B}_i \leftarrow \mathbb{Z}_q^{n\times m}\}$$

Therefore, we can conclude that

$$\left\{(x_i, x_i \cdot \mathbf{B}_i) \ : \ \begin{array}{l} x_i \leftarrow \{-1,1\}, \\ (\mathbf{B}_i, T_{\mathbf{B}_i}) \leftarrow \mathsf{TrapGen}(1^n) \end{array}\right\} \approx \left\{(x_i, \mathbf{B}_i) \ : \ \begin{array}{l} x_i \leftarrow \{-1,1\}, \\ \mathbf{B}_i \leftarrow \mathbb{Z}_q^{n\times m} \end{array}\right\}.$$

As a result, the public key distributions in Hyb_0 and Hyb_1 are statistically indistinguishable. ∎

Claim 4.3. For any adversary $\mathcal{A}$, $\mathsf{Adv}^1_{\mathcal{A}} - \mathsf{Adv}^2_{\mathcal{A}} \leq \text{negl}(n)$.

Proof. The only difference between hybrid experiments Hyb_1 and Hyb_2 is in the choice of $\mathbf{h}_\ell$. In Hyb_1, $\mathbf{h}_\ell = -\sum_i x_i \mathbf{h}_i$, while in Hyb_2, it is chosen uniformly at random. Here, we will use the Leftover Hash Lemma (Theorem 2.1). Since $\ell > (n+1)\log_2 q + \omega(\log n)$, it follows that

$$\{(\mathbf{A} = [\mathbf{h}_1|\ldots|\mathbf{h}_{\ell-1}], \mathbf{h}_\ell = -\mathbf{A}\cdot\mathbf{r}) \ : \ \mathbf{h}_i \leftarrow \mathbb{Z}_q^n \text{ for all } i \leq \ell-1, \mathbf{r} \leftarrow \mathbb{Z}_q^{\ell-1}\}$$
$$\approx$$
$$\{(\mathbf{A} = [\mathbf{h}_1|\ldots|\mathbf{h}_{\ell-1}], \mathbf{h}_\ell) \ : \ \mathbf{h}_i \leftarrow \mathbb{Z}_q^n \text{ for all } i \leq \ell\}$$

∎

Claim 4.4. Assuming (n, ℓ, q, χ)-$\mathsf{LWE\text{-}ss}$ (Assumption 2), for any PPT adversary A, $\mathsf{Adv}^2_{\mathcal{A}} - \mathsf{Adv}^3_{\mathcal{A}} \leq \text{negl}(n)$.

Proof. The only difference in Hyb_2 and Hyb_3 is the manner in which $\mathbf{f}_i$ are computed. In Hyb_2, the challenger chooses $\mathbf{C} \leftarrow \chi^{n\times n}$, $\mathbf{e}_i \leftarrow \chi^n$ and sets $\mathbf{f}_i = \mathbf{C}\cdot\mathbf{h}_i + \mathbf{e}_i$ for all $i \leq \ell$. In Hyb_3, $\mathbf{f}_i$ are chosen uniformly at random from $\mathbb{Z}_q^n$.

Suppose there exists an adversary $\mathcal{A}$ such that $\mathsf{Adv}^2_{\mathcal{A}} - \mathsf{Adv}^3_{\mathcal{A}}$ is non-negligible in n. Then there exists a reduction algorithm $\mathcal{B}$ that can use $\mathcal{A}$ to break Assumption 2 with non-negligible advantage. First, $\mathcal{B}$ receives as LWE challenge two $n\times\ell$ matrices $(\mathbf{H}, \mathbf{F})$. It chooses ℓ matrices $\mathbf{B}_i \leftarrow \mathbb{Z}_q^{n\times m}$, sets $\mathbf{h}_i$ as the i^{th} column of $\mathbf{H}$ and sends $\{\mathbf{B}_i, \mathbf{h}_i\}$ as the public key.

On receiving the challenge messages $\mathsf{msg}_0, \mathsf{msg}_1$, $\mathcal{B}$ uses $\mathbf{F}$ to compute the challenge ciphertext. It first chooses $b \leftarrow \{0,1\}$, computes $(\mathbf{Z}_i, T_{\mathbf{Z}_i})$ using msg_b and sets $\mathbf{f}_i$ to be the i^{th} column of $\mathbf{F}$. Next, it computes $\mathbf{s}_i \leftarrow \mathsf{SamplePre}(\mathbf{Z}_i, T_{\mathbf{Z}_i}, \sigma, \mathbf{f}_i)$ and sends the vectors $\{\mathbf{s}_i\}$ as the ciphertext. Finally, the adversary sends the guess b'. If $b = b'$, $\mathcal{B}$ guesses that $\mathbf{F}$ is an LWE matrix, else it guesses that $\mathbf{F}$ is uniformly random.

Clearly, if $\mathbf{F} = \mathbf{C}\cdot\mathbf{H} + \mathbf{E}$ for some $\mathbf{C} \leftarrow \chi^{n\times n}$, $\mathbf{E} \leftarrow \chi^{n\times\ell}$, then B simulates Hyb_2, and if $\mathbf{F}$ is uniformly random, then this corresponsds to Hyb_3. This concludes our proof. ∎

Claim 4.5. Assuming the well-distributedness property of $(\mathsf{TrapGen}, \mathsf{SamplePre})$ 2.2, for any adversary $\mathcal{A}$, $\mathsf{Adv}^3_{\mathcal{A}} - \mathsf{Adv}^4_{\mathcal{A}} \leq \text{negl}(n)$.

Proof. This follows directly from the well-distributedness property of $(\mathsf{TrapGen}, \mathsf{SamplePre})$ algorithms, because the vectors $\{\mathbf{f}_i\}_i$ are chosen uniformly at random from $\mathbb{Z}_q^n$. Therefore, the well-distributedness property states that for all random coins $\mathbf{y}$, $\{\mathbf{s}_i : (\mathbf{M}, T_{\mathbf{M}}) \leftarrow \mathsf{TrapGen}(1^n; \mathbf{y}), \mathbf{s}_i \leftarrow \mathsf{SamplePre}(\mathbf{M}, T_{\mathbf{M}}, \sigma, \mathbf{f}_i)\} \approx_s \mathcal{D}_{\mathbb{Z}^m,\sigma}$. ∎

Using the above claims, we can show that $\mathsf{Adv}^0_{\mathcal{A}} - \mathsf{Adv}^5_{\mathcal{A}} \leq \text{negl}(n)$, and therefore, the scheme is IND-CPA secure for Leader setup.

4.4 INDCPA Security for Follower Mode

This case is similar to the Leader mode, therefore we will only describe the intermediate hybrids, and refer to the corresponding proofs from the section above.

Hyb_0**:** This corresponds to the IND-CPA security game.

1. Setup Phase:
   (a) The challenger first chooses $\mathbf{y}_i \leftarrow \{0,1\}^{\ell_{\text{TG}}}$ for $i \leq \ell$. Next, it computes $(\mathbf{B}_i, T_{\mathbf{B}_i}) = \mathsf{TrapGen}(1^n; \mathbf{y}_i)$. The challenger sends $\{\mathbf{B}_i\}_i$ to the adversary.
2. Challenge Phase
   (a) The adversary sends two messages $\mathsf{msg}_0, \mathsf{msg}_1$. The challenger first chooses $\mathbf{C} \leftarrow \chi^{n\times n}$, $\mathbf{E}_i \leftarrow \chi^{n\times m}$ and sets $\mathbf{F}_i = \mathbf{C}\cdot\mathbf{B}_i + \mathbf{E}_i$.
   (b) Next, it chooses $b \leftarrow \{0,1\}$. Let $\mathsf{msg}_b = (\mathbf{m}_1, \ldots, \mathbf{m}_\ell)$. The challenger computes $(\mathbf{Z}_i, T_{\mathbf{Z}_i}) = \mathsf{TrapGen}(1^n; \mathbf{m}_i)$.
   (c) Using $T_{\mathbf{Z}_i}$, the challenger computes $\mathbf{S}_i \leftarrow \mathsf{SamplePre}(\mathbf{Z}_i, T_{\mathbf{Z}_i}, \sigma, \mathbf{F}_i)$ for all $i \leq \ell$. It sends $\mathsf{ct}^* = \{\mathbf{S}_i\}_i$ as the challenge ciphertext.
3. Guess: The adversary sends its guess b' and wins if $b = b'$.

Hyb_1**:** In this hybrid, the challenger uses truly random matrices $\mathbf{B}_i$.

1. Setup Phase:
   (a) The challenger chooses $\mathbf{B}_i \leftarrow \mathbb{Z}_q^{n\times m}$ for $i \leq \ell$ and sends $\{\mathbf{B}_i\}_i$ to the adversary.
2. Challenge Phase
   (a) The adversary sends two messages $\mathsf{msg}_0, \mathsf{msg}_1$. The challenger first chooses $\mathbf{C} \leftarrow \chi^{n\times n}$, $\mathbf{E}_i \leftarrow \chi^{n\times m}$ and sets $\mathbf{F}_i = \mathbf{C}\cdot\mathbf{B}_i + \mathbf{E}_i$.
   (b) Next, it chooses $b \leftarrow \{0,1\}$. Let $\mathsf{msg}_b = (\mathbf{m}_1, \ldots, \mathbf{m}_\ell)$. The challenger computes $(\mathbf{Z}_i, T_{\mathbf{Z}_i}) = \mathsf{TrapGen}(1^n; \mathbf{m}_i)$.
   (c) Using $T_{\mathbf{Z}_i}$, the challenger computes $\mathbf{S}_i \leftarrow \mathsf{SamplePre}(\mathbf{Z}_i, T_{\mathbf{Z}_i}, \sigma, \mathbf{F}_i)$ for all $i \leq \ell$. It sends $\mathsf{ct}^* = \{\mathbf{S}_i\}_i$ as the challenge ciphertext.
3. Guess: The adversary sends its guess b' and wins if $b = b'$.

Hyb_2: In this hybrid, the challenger uses truly random matrices $\mathbf{F}_i$ to compute the ciphertext.

1. Setup Phase:
   (a) The challenger chooses $\mathbf{B}_i \leftarrow \mathbb{Z}_q^{n\times m}$ for $i \le \ell$ and sends $\{\mathbf{B}_i\}_i$ to the adversary.
2. Challenge Phase
   (a) The adversary sends two messages $\mathsf{msg}_0, \mathsf{msg}_1$. The challenger first chooses $\underline{\mathbf{F}_i \leftarrow \mathbb{Z}_q^{n\times m}}$ for all $i \le \ell$.
   (b) Next, it chooses $b \leftarrow \{0,1\}$. Let $\mathsf{msg}_b = (\mathbf{m}_1,\ldots,\mathbf{m}_\ell)$. The challenger computes $(\mathbf{Z}_i, T_{\mathbf{Z}_i}) = \mathsf{TrapGen}(1^n; \mathbf{m}_i)$.
   (c) Using $T_{\mathbf{Z}_i}$, the challenger computes $\mathbf{S}_i \leftarrow \mathsf{SamplePre}(\mathbf{Z}_i, T_{\mathbf{Z}_i}, \sigma, \mathbf{F}_i)$ for all $i \le \ell$. It sends $\mathsf{ct}^* = \{\mathbf{S}_i\}_i$ as the challenge ciphertext.
3. Guess: The adversary sends its guess b' and wins if $b = b'$.

Hyb_3: In this hybrid, the challenger chooses the matrices $\mathbf{S}_i$ with entries from the discrete Gaussian distribution $\mathcal{D}^m_{\mathbb{Z}^m,\sigma}$. Therefore, in this game, any adversary has 0 advantage.

1. Setup Phase:
   (a) The challenger chooses $\mathbf{B}_i \leftarrow \mathbb{Z}_q^{n\times m}$ for $i \le \ell$ and sends $\{\mathbf{B}_i\}_i$ to the adversary.
2. Challenge Phase
   (a) The adversary sends two messages $\mathsf{msg}_0, \mathsf{msg}_1$. The challenger first chooses $\mathbf{F}_i \leftarrow \mathbb{Z}_q^{n\times m}$ for all $i \le \ell$.
   (b) Next, it chooses $\underline{\mathbf{S}_i \leftarrow \mathcal{D}^m_{\mathbb{Z}^m,\sigma}}$ for all $i \le \ell$. It sends $\mathsf{ct}^* = \{\mathbf{S}_i\}_i$.
3. Guess: The adversary sends its guess b' and wins if $b = b'$.

Analysis: As mentioned above, the proofs for this section will be very similar to the ones in Section 4.3.1.

Claim 4.6. For any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^0_{\mathcal{A}} - \mathsf{Adv}^1_{\mathcal{A}} \le \mathrm{negl}(n)$.

The proof of this claim is identical to the proof of Claim 4.2.

Claim 4.7. Assuming $(n, m\cdot\ell, q, \chi)$-$\mathsf{LWE\text{-}ss}$ (Assumption 2), for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^1_{\mathcal{A}} - \mathsf{Adv}^2_{\mathcal{A}} \le \mathrm{negl}(n)$.

The proof of this claim is similar to the proof of Claim 4.4.

Claim 4.8. Assuming the well-distributedness property of $(\mathsf{SamplePre}, \mathsf{TrapGen})$ algorithms (Definition 2.2), for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^3_{\mathcal{A}} - \mathsf{Adv}^4_{\mathcal{A}} \le \mathrm{negl}(n)$.

This proof is identical to the proof of Claim 4.5.

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. Cryptology ePrint Archive, Report 2016/110, 2016. `http://eprint.iacr.org/`.

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

[8] 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.

[9] John Black, Phillip Rogaway, and Thomas Shrimpton. Encryption-scheme security in the presence of key-dependent messages. In *Selected Areas in Cryptography, 9th Annual International Workshop, SAC 2002, St. John's, Newfoundland, Canada, August 15-16, 2002. Revised Papers*, pages 62–75, 2002.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

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

[16] 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.

[17] 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.

[18] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT*, 2013.

[19] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, 2013.

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

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

[22] 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.

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

[24] 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.

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

[26] 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.

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

[28] 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.