New Circular Security Counterexamples from Decision Linear and Learning with Errors

Allison Bishop*
Columbia University
allison@cs.columbia.edu

Susan Hohenberger†
Johns Hopkins University
susan@cs.jhu.edu

Brent Waters‡
University of Texas at Austin
bwaters@cs.utexas.edu

November 30, 2015

Abstract

We investigate new constructions of n-circular counterexamples with a focus on the case of $n = 2$. We have a particular interest in what qualities a cryptosystem must have to be able to separate such circular security from IND-CPA or IND-CCA security. To start, we ask whether there is something special about the asymmetry in bilinear groups that is inherent in the works of [1] and [17] or whether it is actually the bilinearity that matters. As a further question, we explore whether such counterexamples are derivable from other assumptions such as the Learning with Errors (LWE) problem. If it were difficult to find such counterexamples, this might bolster our confidence in using 2-circular encryption as a method of bootstrapping Fully Homomorphic Encryption systems that are based on lattice assumptions.

The results of this paper broadly expand the class of assumptions under which we can build 2-circular counterexamples. We first show for any constant $k \geq 2$ how to build counterexamples from a bilinear group under the decision k-linear assumption. Recall that the decision k-linear assumption becomes progressively weaker as k becomes larger. This means that we can instantiate counterexamples from symmetric bilinear groups and shows that asymmetric groups do not have any inherently special property needed for this problem. We then show how to create 2-circular counterexamples from the Learning with Errors problem. This extends the reach of these systems beyond bilinear groups and obfuscation.

1 Introduction

The notion of key dependent message security [11] moves beyond our classical notion of encryption security [21]. It demands a system remain secure even if an attacker gains access to ciphertexts that encrypt messages that are, or depend on, the very private keys of the system it is trying to attack. As a concrete example, consider a special case of key-dependent security called n-circular security. Here an encryption scheme is said to be n-circular secure, if an adversary is unable to distinguish $\mathsf{Enc}(pk_1, sk_2), \mathsf{Enc}(pk_2, sk_3), \ldots, \mathsf{Enc}(pk_n, sk_1)$ from corresponding zero encryptions.

While the notion of key dependent or circular security might first appear to be just a technical exercise, this very problem arises in multiple contexts. Camenisch and Lysyanskaya [16] applied circular secure

*Supported by NSF CNS 1413971 and NSF CCF 1423306.

†Supported by the National Science Foundation (NSF) CNS-1228443 and CNS-1414023, the Office of Naval Research under contract N00014-14-1-0333, and a Microsoft Faculty Fellowship.

‡Supported by NSF CNS-1228599 and CNS-1414082, DARPA SafeWare, a Google Faculty Research Award, the Alfred P. Sloan Fellowship, Microsoft Faculty Fellowship, and Packard Foundation Fellowship.

encryption to build an anonymous credentials scheme with certain properties. Other works used circular security in formal methods to prove the soundness of symbolic protocols [2, 25]. Perhaps the most compelling example comes from Gentry [19], who showed that a fully homomorphic scheme for limited depth can be "bootstrapped" to work for arbitrary depth circuits if the original system is sufficient to compute its own decryption circuit and is 1-circular secure.

The first positive examples of key-dependent message security were given in the random oracle model by Black et al. [11] and Camenisch and Lysyanskaya [16]. It was a significant time later when Boneh, Hamburg, Halevi and Ostrovsky [13] gave an elegant construction of an n-circular secure encryption in the standard model under the decision Diffie-Hellman assumption. Subsequently, a sequence of further works [8, 14, 9, 15, 7, 5] gave standard model constructions of key dependent security for functions that could be arbitrary circuits on the private key(s).

All the above constructions and proofs were based on encryption schemes with specific properties. A natural question is whether key-dependent message security is implied by IND-CPA (or IND-CCA) security. If this were true, we would get it for free, without needing such specific properties of the encryption scheme.

A cursory examination of the problem shows that in the broadest sense the answer is no. One can derive a simple counterexample for 1-circular security (i.e., a system that encrypts its own private key) by slightly modifying a public key encryption system. To do so, simply augment a standard private key K with a randomly chosen $K' \in \{0,1\}^\lambda$ and append $y = f(K')$ to the public key where f is a one way function. When encrypting a message $m = (m_1, m_2)$ the system will give out the message in the clear if $f(m_2) = y$) and encrypt normally otherwise. Clearly, an encryption of the private key will be detectable. Yet, if the function f is one way and the original system is IND-CPA secure, the resulting system will still be IND-CPA secure.

While it can be trivially shown (by the argument above) that IND-CPA security does not imply 1-circular security, the case for $n \geq 2$ becomes significantly more challenging. Intuitively, when multiple public keys are thrown into the mix, we need a system that is powerful enough to allow for different ciphertexts to "talk" to each other in a manner that allows for cycle detection, but does not compromise IND-CPA security. So far there have been two approaches to this. For the case of $n = 2$, Acar et al. [1] and Cash, Green and Hohenberger [17] showed how to construct a counterexample from a certain class of *asymmetric* bilinear groups.[1] Here there must exist a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ where the decision Diffie-Hellman problem is believed to remain hard respectively within $\mathbb{G}_1$ and within $\mathbb{G}_2$ (this is called the SXDH assumption). A second approach by Koppula, Ramchen and Waters [24] showed a counterexample under the assumption of indistinguishability obfuscation for poly-sized circuits. Independently and concurrently, Marcedone and Orlandi [26] showed this under the stronger assumption of virtual black box obfuscation.

Our Goals and Results In this work, we investigate new constructions of n-circular counterexamples with a focus on the case of $n = 2$. We have a particular interest in what qualities a cryptosystem must have to be able to separate circular security from IND-CPA and IND-CCA security.

To start, we ask whether there is something special about the asymmetry in bilinear groups that is inherent in the works of [1, 17, 33] or whether it is actually more the bilinearity that matters. As a further question, we explore how to derive such counterexamples from other assumptions such as the Learning with Errors (LWE) problem. If it were difficult to find such counterexamples, this might bolster are confidence in using 2-circular encryption as a method of bootstrapping [19] fully homomorphic encryption systems that are based on lattice assumptions.

The results of this paper broadly expand the class of assumptions from which we can build 2-circular counterexamples. We first show for any constant $k \geq 2$ how to build 2-circular counterexamples from a bilinear group under the decision k-linear assumption. Recall that the decision k-linear assumption becomes progressively weaker as k becomes larger. This means that we can instantiate counterexamples from symmetric bilinear groups and shows that asymmetric groups do not have any inherently special property needed for this problem. We then show how to create 2-circular counterexamples from the Learning with Error (LWE)

[1] In a similar vein, Rothblum [33] presented an elegant counterexample for bit-encryption under a generalization of the SXDH assumption applied to multilinear groups.

problem. This extends the reach of these systems beyond bilinear groups and obfuscation, giving us a much broader understanding of circular security and its challenges.

Our Approach We begin by introducing a new abstraction called an n-Cycle Tester that will simplify the process of finding and describing counterexamples by focusing on the core problem. A *cycle tester* consists of four algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Test})$. The algorithms of $\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}$ behave as in a normal encryption scheme with a common trusted setup algorithm, while the Test algorithm will take in an n-tuple of public keys and ciphertexts and detect (with some non-negligible probability) the presence of a cycle. Notably absent is the inclusion of a decryption algorithm. Thus, a tester does not require that ciphertexts be decryptable in the traditional sense — it only matters that the Test algorithm work with some non-negligible probability. We found that relieving the responsibility of providing a system with decryption simplifies our constructions and allows us to focus on the main ideas. The security property required is IND-CPA security (recall that the basic IND-CPA game does not involve a decryption algorithm).

Of course, to obtain a full-fledged counterexample of an encryption system we actually do need to provide an encryption system that decrypts. We show how to generically derive such a counterexample for n-circular encryption by combining a standard IND-CPA secure cryptosystem (of sufficient message length) with a n-cycle tester. The idea is fairly straightforward. The setup algorithm of the counterexample will run the respective setup algorithms of the encryption and cycle tester schemes. The public key is the pair of these public keys and the secret key is the pair of secret keys. To encrypt a message $m = (m_1, m_2)$, first encrypt $m = (m_1, m_2)$ under the regular encryption system, then encrypt just m_2 under the cycle tester. We can now see that: (1) the cycle tester will allow for any key cycle to be detected and (2) the standard encryption scheme can be used for decryption. A simple hybrid argument shows that the IND-CPA security of the standard encryption scheme and cycle tester imply IND-CPA security of the derived counterexample system.

We also show that it is possible to extend this transformation idea to chosen ciphertext security, where we can combine any IND-CCA secure encryption system (of appropriate message length) with the same IND-CPA secure cycle tester to get an encryption system that is IND-CCA secure, but where encryption of key cycles can be detected.

Again, the usefulness of this framework is its modularity. We show these basic transformations once in Section 4, and then for each construction we only need to focus on the basic cycle tester abstraction.

A Cycle Tester from Asymmetric Bilinear Groups As a baseline for our exploration (see auxiliary Section 5 for further details), we first create a 2-cycle tester from asymmetric groups using the SXDH assumption. Our construction is extracted from Cash et al. [17] (also similar to [1, 33]), but simpler in that we only aim for the tester abstraction.

In our construction, the Setup algorithm creates an asymmetric pairing description $\mathrm{PP} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ of prime order p. It also produces generators $g \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$. The message space will be $\mathbb{Z}_p^*$.

A key can be of one of two types. The cycle detection algorithm Test will work on any cycle of keys of two different types. The key generation algorithm KeyGen will first flip a coin $\beta \in \{0, 1\}$ to determine its type. It then picks a random key $s \in \mathbb{Z}_p^*$. If $\beta = 0$, it sets its public key to be $K = g^s \in \mathbb{G}_1$; otherwise, its public key is $K = h^s \in \mathbb{G}_2$.

The encryption algorithm will choose a random exponent $t \in \mathbb{Z}_p$ and if the key is of type $\beta = 0$, it produces the ciphertext as $(C_1 = K^{tm} = g^{stm},\ \ C_2 = g^t) \in \mathbb{G}_1^2$; otherwise if $\beta = 1$, it produces the ciphertext as $(C_1 = K^{tm} = h^{stm},\ \ C_2 = h^t) \in \mathbb{G}_2^2$. With ciphertexts of this form, the test algorithm follows straightforwardly. Suppose we had a pair of ciphertexts $\mathbf{y} = (C = (C_1, C_2), C' = (C_1', C_2'))$ that encrypted a cycle for keys of different types. The algorithm can test this by simply computing $e(C_1, C_2') \stackrel{?}{=} e(C_2, C_1')$. Plugging in s, s' as the respective keys, t, t' as the encryption randomness, and m, m' as the messages, we see that the test computes:

$$e(g^{stm}, h^{t'}) \stackrel{?}{=} e(g^t, h^{s't'm'}).$$

This equality holds if $m = s'$ and $m' = s$ and will not hold with high probability for a message independent of the private key.

One thing we emphasize here is that IND-CPA is clearly broken if the SXDH assumption does not hold. Consider an encryption $(C_1 = K^{tm} = g^{stm}, \quad C_2 = g^t) \in \mathbb{G}_1^2$ for the message m. The group elements $g, (g^s)^m = g^{sm}, C_2 = g^t, C_1 = g^{stm}$ clearly form a DDH tuple. So if DDH is easy in $\mathbb{G}_1$, any $\beta = 0$ type key is susceptible to attack. An analogous statement holds in $\mathbb{G}_2$ for any $\beta = 1$ key. This potential attack demonstrates that the above construction relies strongly on properties of asymmetric groups. We next show how to remove that reliance.

A Cycle Tester from the Decision k-Linear Assumption We next move to constructing a cycle tester from the decision k-linear assumption for any constant $k \geq 2$. Recall that the k-linear assumption [23, 34] is a parameterized family of assumptions on the source elements of bilinear groups. The assumption class becomes progressively weaker for larger values of k. Importantly, by moving to the decision k-linear assumption we remove our dependence on asymmetric groups.[2] See Section 2.1 for a review.

In our construction, the setup algorithm first generates a bilinear source group $\mathbb{G}$ of prime order p with generator g. Then it chooses a random invertible (rank k) matrix $\mathbf{A} \in \mathbb{Z}_p^{k \times k}$ and computes $g^{\mathbf{A}}$, which along with the group description forms the common public parameters. (We use the notation $g^{\mathbf{M}}$ as shorthand for the set of group elements resulting from raising g to each matrix entry in $\mathbf{M}$.) The message and key spaces are defined to be the set of rank k matrices in $\mathbb{Z}_p^{k \times k}$.[3]

Once again the key generation algorithm will flip a coin β to determine its type. Next it chooses a random $\mathbf{W}$ from the set of invertible matrices in $\mathbb{Z}_p^{k \times k}$. If $\beta = 0$ the key is $g^{\mathbf{AW}}$; otherwise it is $g^{\mathbf{WA}}$.

The encryption algorithm takes as input a message $\mathbf{M} \in \mathbb{Z}_p^{k \times k}$ and then computes its inverse $\mathbf{M}^{-1}$. (Recall the message space is the set of invertible matrices.) If the type bit $\beta = 0$, the algorithm chooses a random row vector $\mathbf{r}$ of length k in $\mathbb{Z}_p$ (i.e. a random matrix of dimension $1 \times k$). The ciphertext is computed and output as $C_1 = g^{\mathbf{rAW}}, \quad C_2 = g^{\mathbf{rAM}^{-1}}$. Thus, the ciphertext will consist of two row vectors in the exponent. We observe all terms are computable from the public keys and public parameters. If the type bit $\beta = 1$ the algorithm chooses a random column vector $\mathbf{r}$ of length k in $\mathbb{Z}_p$ (i.e., a random matrix of dimension $k \times 1$). The ciphertext is computed and output as $C_1 = g^{\mathbf{WAr}}, \quad C_2 = g^{\mathbf{M}^{-1}\mathbf{Ar}}$.

Now suppose we have two ciphertexts $\mathbf{y} = (C = (C_1, C_2), \ C' = (C_1', C_2'))$ of different types (with the first being of $\beta = 0$). We can then test for a cycle by testing if $e(C_1, C_2') \stackrel{?}{=} e(C_1', C_2)$. To see why, suppose we had a cycle, so we have that $\mathbf{M}'^{-1} = \mathbf{W}^{-1}$ and $\mathbf{M}^{-1} = \mathbf{W}'^{-1}$. Then, in the exponent, it follows that:

$$\begin{aligned} \mathbf{rAWM}'^{-1}\mathbf{Ar}' &\stackrel{?}{=} \mathbf{rAM}^{-1}\mathbf{W}'\mathbf{Ar}' \\ \mathbf{rAIAr}' &\stackrel{?}{=} \mathbf{rAIAr}' \\ \mathbf{rA}^2\mathbf{r}' &\stackrel{?}{=} \mathbf{rA}^2\mathbf{r}'. \end{aligned}$$

So if there is a cycle, the test will output 1. In contrast, if the messages encrypted are independent of the key, the test will output 0 with high probability.

Finally, we can give a simple proof of IND-CPA security from the decision k-linear assumption. More specifically, we will use the matrix k-linear assumption, introduced by Naor and Segev [28], that was shown to be equivalent to the decision k-linear assumption. Informally, the assumption says that it is hard to distinguish $g^{\mathbf{X}}$ and $g^{\mathbf{Y}}$ where $\mathbf{X}$ is a random matrix of rank $i > k$ and $\mathbf{Y}$ is a random matrix (of the same dimension) of rank $j > k$. I.e., the rank of matrices in the exponent cannot be determined as long as it is greater than k. For our purposes, we will be interested in using the difficulty of distinguishing between rank k and rank $k + 1$ matrices.

[2] We emphasize though that our constructions could use an asymmetric form of bilinear maps if desired, although we describe things in terms of symmetric groups. The main point is that there is no longer a reliance on asymmetry or that DDH is hard within each group.

[3] In our scheme, we actually let the message and key space be $\{0,1\}^\lambda$ for security parameter λ and define a pseudorandom generator from this to rank k matrices. That way the message space is defined before the common setup is executed. However, for simplicity we will just assume here that the message and key spaces are the set of invertible $k \times k$ matrices.

Let us examine IND-CPA security for an encryption under a type $\beta = 0$ key. (The argument for $\beta = 1$ will follow analogously.) We will devise a reduction algorithm that receives a matrix k-linear assumption challenge $g^{\mathbf{M}}$, where $\mathbf{M}$ is selected as either a random rank k matrix or rank $k+1$ matrix. In the case where it is a rank k matrix, our reduction algorithm will use it to derive the key and ciphertext values of

$$g^{\mathbf{A}}, g^{\mathbf{AW}}, g^{\mathbf{rAW}}, g^{\mathbf{rA}}.$$

These can be used to generate a well-formed ciphertext of a given message. However, if the reduction algorithm receives a random matrix of rank $k+1$, it will create key and ciphertext values distributed as

$$g^{\mathbf{A}}, g^{\mathbf{AW}}, g^{\mathbf{rAW}}, g^{\mathbf{uA}}.$$

In this case the fact that $\mathbf{u}$ is fresh randomness will information-theoretically hide the message from the attacker. It then follows that any attacker with non-negligible advantage against our system must break the matrix k-linear assumption.

In auxiliary Section A, we present a different 2-cycle tester from the Decision Linear assumption in symmetric pairing groups. This construction can be viewed as closer to an extension of the SXDH one from auxiliary Section 5 to symmetric groups where new variables and equations are introduced to prevent the use of pairings to disrupt IND-CPA security. However, it does not seem to generalize to a system that is secure using the decision k-linear assumption for $k > 2$ or help move toward a Learning with Errors Assumption. At the same time, when compared to our more general construction just given for the $k = 2$ (decision linear assumption) case, it achieves smaller public keys. Public keys here are two group elements as opposed to four. Our techniques for this construction might be of future interest for other applications of transforming constructions proved under asymmetric group assumptions to those that do not rely on them. We defer further details of these techniques to auxiliary Section A.

A Cycle Tester from Learning with Errors Assumption While there are now many known examples of cryptographic functionalities that can be achieved in both the bilinear and lattice settings, it is not at all clear how to imitate the pairings-based approach above to obtain a cycle tester from the LWE assumption. Typically, encryption schemes proven secure under LWE have ciphertexts that are large, noisy vectors in $\mathbb{Z}_q^m$ and secret keys that are short vectors in $\mathbb{Z}^m$, with decryption computing a dot product and then removing the small effect of the noise multiplied by the short key vector. It seems unlikely that we could build a cycle tester using only this kind of structure, as the cycle effect would be obscured by the interactions of large ciphertext vectors with the embedded noise.

Intuitively, we then expect that a cycle tester may use ciphertexts that have two parts: a noisy vector and a short vector. The large, noisy vectors will help us prove IND-CPA security from LWE, while the short vectors will help us perform the cycle test. Naturally, the main challenge is designing the relationship between the noisy and short vectors such that the short vectors do not break security when there is no cycle.

The secret key for our scheme will generate a matrix B and a corresponding short trapdoor basis T_B. For IND-CPA security, it is important that B is hidden, so one should ignore the notational collision and not think of this as corresponding to the public matrix A in an LWE challenge, but rather the columns of B will play the role of different hidden s vectors in typical LWE notation. The public key will be formed by choosing several random vectors $c_1, \ldots, c_\ell$ and publishing noisy versions of $c_1 B, \ldots, c_\ell B$ as well as the (non-noisy) vectors $c_1, \ldots, c_\ell$ (so these c_i's can be thought of as playing the role of the public matrix A in an LWE challenge).

To encrypt a message, the message will first be used to generate a matrix Z and a corresponding short trapdoor basis T_Z. The encryptor will mimic typical LWE-style encryption by forming a noisy version of sB for some vector s, but since it does not know B, it will form s as a linear combination of $c_1, \ldots, c_\ell$ with coefficients chosen randomly from $\{-1, 1\}$. Note that the encryptor can then compute both s (without noise) and a noisy version of sB. The noisy version of sB becomes the noisy part of the ciphertext, and the other part of the ciphertext is a short vector v such that Zv equals the transpose of s. Note that such a vector v can be sampled appropriately using the trapdoor basis T_Z.

For full details of how the cycle test works, see Section 7. The main idea is that when there is a 2-cycle, the secret key matrix B for one ciphertext is the same as the message matrix Z for the other ciphertext and vice versa. This leads to a common relationship between the short vector of one ciphertext and the noisy vector of the other, while when the B, Z matrices of each are fresh and unrelated, this relationship does not appear. One convenient feature of this scheme as compared to the bilinear schemes is that there is no need for different types of ciphertexts. Intuitively, the pairing relationship has been replaced by a dot product relationship between a short vector and a noisy one.

Proving IND-CPA security for this scheme can be accomplished in a few steps. First, since B is hidden and its columns act like the hidden vector s in a typical LWE challenge and the c_i's act like rows of the public matrix A, we can argue that LWE implies the noisy public versions of $c_i B$ can be replaced by uniformly random vectors, independent of the c_i's and B. Next, using a convenient variant of the left over hash lemma from [3], we argue that the random coefficients in $\{-1, 1\}$ that form s from the c_i's and the noisy ciphertext vector from the public noisy vectors supply sufficient entropy to replace both of these with fresh uniformly random vectors as well. We are then left with an encryption that samples a uniformly random s (now independent of the noisy part of the ciphertext) and samples the short part of the ciphertext as a short vector v such that Zv is the transpose of s. Here we can argue that the distribution of such a v is statistically close to a distribution that is independent of Z: this follows from a result in [20] that ensures us that the image of a short, Gaussian distributed vector v under multiplication by Z is uniformly distributed in $\mathbb{Z}_q^n$. Thus, by employing LWE followed by a sequence of statistical arguments, we can arrive at a point where the ciphertext is independent of the message, and this implies IND-CPA security.

Other Related Work Haitner and Holenstein [22] show black box impossibility results for proving key-dependent message security from different cryptographic assumptions. Their goal deviates from ours in two important ways. First, their work focuses on impossibility results for ciphertext encrypting functions of its own private keys, whereas we are concerned with the circular case where there is a cycle over multiple private keys. Second, we are interested in concrete counterexamples. In particular, it may be possible that IND-CPA security implies certain key-dependent security properties even if there does not exist any black box reduction. In contrast our counterexamples will show that this is impossible if certain specific number theoretic assumptions hold.

2 Preliminaries

2.1 Pairings

Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be groups of order p. A map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an admissible *pairing* (also called a *bilinear map*) if it satisfies the following three properties:

1. Bilinearity: for all $g \in \mathbb{G}_1$, $h \in \mathbb{G}_2$, and $a, b \in \mathbb{Z}_p$, it holds that $e(g^a, h^b) = e(g^b, h^a) = e(g, h)^{ab}$.
2. Non-degeneracy: if g and h are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, resp., then $e(g, h)$ is a generator of $\mathbb{G}_T$.
3. Efficiency: there exists an efficient method that given any $g \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$, computes $e(g, h)$.

A pairing generator $\mathcal{G}$ is an algorithm that on input a security parameter 1^λ, outputs the parameters for a pairing group $(p, g, h, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ such that p is an integer in $\Theta(2^\lambda)$, $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are groups of order p where g generates $\mathbb{G}_1$, h generates $\mathbb{G}_2$ and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an admissible pairing.

The above pairing is called an *asymmetric* or Type-III pairing. This type of pairing is generally preferred in implementations for its efficiency. However, many works use *symmetric* or Type-I pairings, where there is an efficient isomorphism $\psi : \mathbb{G}_1 \to \mathbb{G}_2$ (and vice versa) such that a symmetric map is defined as $e : \mathbb{G}_1 \times \psi(\mathbb{G}_1) \to \mathbb{G}_T$. In this setting, we generally treat $\mathbb{G} = \mathbb{G}_1 = \mathbb{G}_2$ for simplicity and write $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. As we discuss shortly, Type-I pairings may allow for *weaker* complexity assumptions.

Decisional Diffie-Hellman Assumption (DDH) Let $\mathbb{G}$ be a group of prime order $p \in \Theta(2^\lambda)$. For all p.p.t. adversaries $\mathcal{A}$, the following probability is $1/2$ plus an amount negligible in λ:

$$\Pr[g \leftarrow \mathbb{G}; a, b \leftarrow \mathbb{Z}_p^*; T_0 = g^{ab}; T_1 \leftarrow \mathbb{G}; d \leftarrow \{0,1\}; d' \leftarrow \mathcal{A}(g, g^a, g^b, T_d) : d = d'].$$

Strong External Diffie-Hellman Assumption (SXDH): Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be bilinear groups. The SXDH assumption states that the Decisional Diffie-Hellman (DDH) problem is hard in both $\mathbb{G}_1$ and in $\mathbb{G}_2$. This implies that there does *not* exist an efficiently computable isomorphism between these two groups, so SXDH is only possible in the asymmetric or Type-III setting.

Decision Linear and the k-LIN Family (k-LIN) We now present a family of assumptions called the k-LIN assumptions (where $k = 1$ is the standard DDH assumption and $k = 2$ is called Decision Linear [12]) [23, 10]. Let $\mathbb{G}$ be a group of prime order $p \in \Theta(2^\lambda)$. For all p.p.t. adversaries $\mathcal{A}$ and $k \geq 1$, the following probability is $1/2$ plus an amount negligible in λ:

$$\Pr[g, g_1, \ldots, g_k \leftarrow \mathbb{G}; r_1, \ldots, r_k \leftarrow \mathbb{Z}_p; T_0 = g^{(r_1+\cdots+r_k)}; T_1 \leftarrow \mathbb{G}; d \leftarrow \{0,1\}; \\ d' \leftarrow \mathcal{A}(g, g_1, \ldots, g_k, g_1^{r_1}, \ldots, g_k^{r_k}, T_d) : d = d'].$$

In the generic group model, these k-LIN assumptions become progressively weaker for increasing k.

In our proof of security in Section 6 we will use a theorem due to Naor and Segev [28] that shows that under the decision k-linear assumption no attacker can distinguish between a random rank i matrix and a random rank j matrix (in the exponent and of the same dimensions) for $i, j \geq k$.

2.2 Lattices and LWE

We let q, n, and m denote positive integers. Given 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 $\{x \in \mathbb{Z}^m \ : \ Ax = u \mod q\}$.

For a matrix $A \in \mathbb{Z}^{n \times m}$, we let $||A||$ denote the ℓ_2 length of the longest column of A, and we let $||A||_{GS}$ denote $||\widetilde{A}||$, where $\widetilde{A}$ is the Gram-Schmidt orthogonalization of the columns of A. We let A^t denote the transpose of the matrix A.

Learning with Errors (LWE) Given integers n, m, a prime q, and a noise distribution χ over $\mathbb{Z}$, the (n, m, q, χ)-LWE problem is to distinguish the distributions $(A, A^t s + e)$ and (A, u), where A is chosen uniformly from $\mathbb{Z}_q^{n \times m}$, s is chosen uniformly from $\mathbb{Z}_q^n$, e is chosen from χ^m, and u is chosen uniformly from $\mathbb{Z}_q^m$.

Under a quantum reduction, Regev [32] showed that for certain noise distributions, the LWE problem is as hard as the worst-case SIVP and GapSVP. Peikert [30] gave a reduction in the classical setting. Our construction will admit a range of parameters where solving the LWE problem is as hard as approximating the worst-case GapSVP to polynomial (in n) factors, which is believed to be computationally hard.

Trapdoor Generation We will rely on the polynomial time algorithm $\mathrm{TrapGen}(1^n, 1^m, q)$ (developed in [4, 6, 27]). This is a randomized algorithm that when given $m = \Theta(n \log q)$, outputs a full rank matrix $A \in \mathbb{Z}_q^{n \times m}$ and an accompanying basis $T_A \in \mathbb{Z}^{m \times m}$ for $\Lambda_q^\perp(A)$ such that the distribution of A is negligibly close (in n) to uniform over $\mathbb{Z}_q^{n \times m}$ and $||T_A||_{GS} = \mathcal{O}(\sqrt{n \log q})$ with probability 1 (this can be obtained from [27]).

Discrete Gaussian Distributions We employ the discrete Gaussian distribution $\mathcal{D}_\sigma(\Gamma_q^u(A))$ on $\Gamma_q^u(A)$, parameterized by $\sigma > 0$ (as defined e.g. in [32]). The salient fact we will use about this distribution is that for a random matrix $A \in Z_q^{n \times m}$ and $\sigma = \tilde{\Omega}(\sqrt{n})$, a vector sampled from $\mathcal{D}_\sigma(\Lambda_q^u(A))$ has ℓ_2 norm less than $\sigma\sqrt{m}$ with probability at least 1 minus a quantity that is negligible in m.

We will rely on a polynomial time algorithm SampleD(A, T_A, u, σ) [20]. This is a randomized algorithm that when $\sigma = ||T_A||_{GS} \cdot \omega(\sqrt{\log m})$, produces a random vector x from a distribution that is statistically close to $\mathcal{D}_\sigma(\Lambda_q^u(A))$.

We also employ the following result from [20] (appears as Corollary 5.4 in that work):

Lemma 2.1 *Let n and q be positive integers with q prime, and let $m \geq 2n \log q$. Then for all but a $2q^{-n}$ fraction of all $A \in \mathbb{Z}_q^{n \times m}$ and for any $\sigma \geq \omega(\sqrt{\log m})$, the distribution of the syndrome $u = Ae \mod q$ is statistically close to uniform over $\mathbb{Z}_q^n$, where e is distributed according to $\mathcal{D}_{\mathbb{Z}^m, \sigma}$.*

Randomness Extraction We will use the leftover hash lemma (see [3] e.g. for an even stronger statement):

Lemma 2.2 *Suppose that $\ell > (j+1)\log q + \omega(\log j)$ and $q > 2$ is prime (for integers q, j, ℓ). Let R be an $\ell \times 1$ vector chosen uniformly in $\{1, -1\}^\ell \mod q$. Let A and B be matrices chosen uniformly in $\mathbb{Z}_q^{j \times \ell}$ and $\mathbb{Z}_q^{j \times 1}$ respectively. Then, the distribution (A, AR) is statistically close to the distribution (A, B).*

3 Security Definitions

In this work, we will focus on public key encryption schemes that admit a global setup algorithm.

Definition 3.1 (Public Key Encryption) *A* public key encryption scheme $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *for a message space M and secret key space S[4] is a tuple of algorithms specified as follows:*

- $\mathsf{Setup}(1^\lambda) \to \mathrm{PP}$. *The Setup algorithm takes as input the security parameter λ and outputs common public parameters* PP.
- $\mathsf{KeyGen}(\mathrm{PP}) \to (pk, sk)$. *The Key Generation algorithm takes as input the public parameters* PP *and outputs a public pk and secret key $sk \in S$.*
- $\mathsf{Enc}(pk, m \in M) \to C$. *The Encryption algorithm takes as input a public key pk and a message $m \in M$ and outputs a ciphertext C.*
- $\mathsf{Dec}(sk, C) \to m$. *The Decryption algorithm takes as input a secret key sk and a ciphertext C and outputs either an error message $\perp$ or a value $m \in M$.*

By $\mathsf{negl}(k)$ we denote some *negligible* function, i.e., one such that, for all $c > 0$ and all sufficiently large k, $\mathsf{negl}(k) < 1/k^c$. We abbreviate probabilistic polynomial time as PPT.

Perfect Correctness. An encryption scheme $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ for message space M is said to be perfectly correct if for all $\lambda \in \mathbb{N}$, $m \in M$, and $(pk, sk) \in \mathsf{KeyGen}(\mathsf{Setup}(1^\lambda))$, it holds that $\mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) = m$.

Security. We recall the notion of indistinguishability of encryptions under a chosen-plaintext attack [21].

Definition 3.2 (IND-CPA Security) *Let $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme. For scheme Π, adversary $\mathcal{A}$, and $\lambda \in \mathbb{N}$, let the random variable $\mathsf{IND\text{-}CPA}(\Pi, \mathcal{A}, \lambda)$ be defined by the probabilistic algorithm described on the left side of Figure 1. We denote the* IND-CPA advantage of $\mathcal{A}$ *by $\mathsf{Adv}^{\mathsf{cpa}}_{\Pi,\mathcal{A}}(\lambda) = 2 \cdot \Pr[\mathsf{IND\text{-}CPA}(\Pi, \mathcal{A}, \lambda) = 1] - 1$. We say that Π is* IND-CPA secure *if $\mathsf{Adv}^{\mathsf{cpa}}_{\Pi,\mathcal{A}}(\lambda)$ is negligible for all PPT $\mathcal{A}$.*

We also consider the indistinguishability of encryptions under a chosen-ciphertext attack [29, 31, 18].

[4] Technically, the output of the Setup algorithm may be required to establish the message and secret key spaces. For instance, the setup algorithm may output a prime p and the message space might be set as $\mathbb{Z}_p^*$. For simplicity, we provide a name for these sets at the scheme level, even though the elements in these sets may not be defined until after Setup.

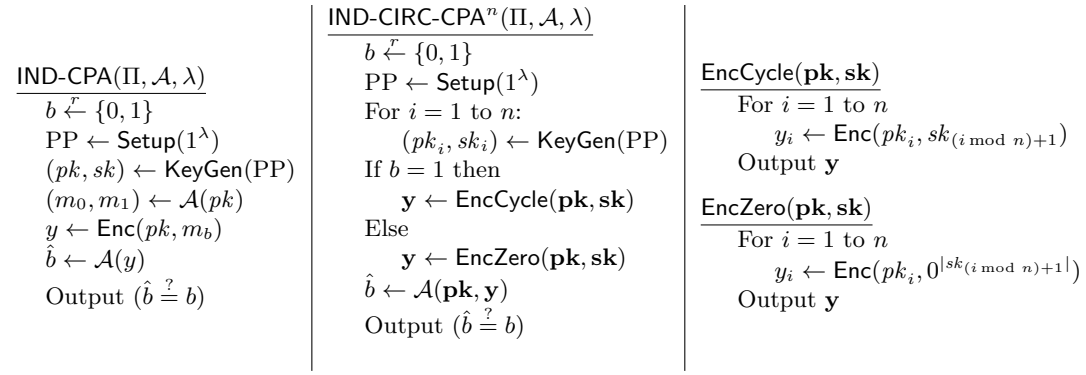

Figure 1: Experiments for Definitions 3.2 and 3.4, each for a message space M, and we assume that $m_0, m_1, sk_i \in M$. We write $\mathbf{pk}$, $\mathbf{sk}$, and $\mathbf{y}$ for $(pk_1, \ldots, pk_n)$, $(sk_1, \ldots, sk_n)$ and $(y_1, \ldots, y_n)$ respectively.

Definition 3.3 (IND-CCA Security) *Let* $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *be a public-key encryption scheme. Let the random variable* $\mathsf{IND\text{-}CCA}(\Pi, \mathcal{A}, \lambda)$ *be defined by an algorithm identical to* $\mathsf{IND\text{-}CPA}(\Pi, \mathcal{A}, \lambda)$ *above, except that* $\mathcal{A}$ *has access to an oracle* $\mathsf{Dec}(sk, \cdot)$ *that returns the output of the decryption algorithm and* $\mathcal{A}$ *cannot query this oracle on input* y*. We denote the* IND-CCA advantage of $\mathcal{A}$ *by* $\mathsf{Adv}^{\mathsf{cca}}_{\Pi,\mathcal{A}}(\lambda) = 2 \cdot \Pr[\mathsf{IND\text{-}CCA}(\Pi, \mathcal{A}, \lambda) = 1] - 1$*. We say that* Π *is* IND-CCA secure *if* $\mathsf{Adv}^{\mathsf{cca}}_{\Pi,\mathcal{A}}(\lambda)$ *is negligible for all PPT* $\mathcal{A}$*.*

3.1 Circular Security

We next define circular security of public-key encryption. This definition is derived from the Key-Dependent Message (KDM) security notion of Black et al. [11]. We follow prior counterexample definitions [1, 17] which restrict the adversary's power (e.g., cannot ask for any affine function of the secret keys). The adversary is asked to distinguish between an encryption cycle or encryptions of zero as in [13, 17]. The bit string zero is not actually in the message spaces we consider, but this value can be encoded to be in the space; equivalently, one can follow the approach of Acar et al. [1] which instead of zero, encrypts a fresh random message.

Definition 3.4 ($\mathsf{IND\text{-}CIRC\text{-}CPA}^n$) *Let* $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *be a public-key encryption scheme. For integer* $n > 0$*, scheme* Π*, adversary* $\mathcal{A}$ *and* $\lambda \in \mathbb{N}$*, let the random variable* $\mathsf{IND\text{-}CIRC\text{-}CPA}^n(\Pi, \mathcal{A}, \lambda)$ *be defined by the probabilistic algorithm in the middle of Figure 1. We denote the* $\mathsf{IND\text{-}CIRC\text{-}CPA}^n$ advantage of $\mathcal{A}$ *by*

$$\mathsf{Adv}^{n\text{-}\mathsf{circ}\text{-}\mathsf{cpa}}_{\Pi,\mathcal{A}}(\lambda) = 2 \cdot \Pr[\mathsf{IND\text{-}CIRC\text{-}CPA}^n(\Pi, \mathcal{A}, \lambda) = 1] - 1.$$

We say that Π *is* $\mathsf{IND\text{-}CIRC\text{-}CPA}^n$ secure *if* $\mathsf{Adv}^{n\text{-}\mathsf{circ}\text{-}\mathsf{cpa}}_{\Pi,\mathcal{A}}(\lambda)$ *is negligible for all PPT* $\mathcal{A}$*.*

Discussion. Cash et al. [17] made a distinction between whether an adversary could *distinguish* an encryption cycle from encryptions of zero (as in the standard game above), or whether an adversary could actually *recover* the secret keys (and provided the latter type of counterexample). Recently, Koppula et al. [24] showed that if there exists (an IND-CPA secure) scheme with a PPT adversary that can distinguish an encryption cycle (in the standard game), then it can be transformed into another scheme with a corresponding adversary that can extract the secret keys from the cycle. Thus, in this work, we can focus exclusively on the standard definition.

4 A Framework for Generating Circular Counterexamples

We now present a general framework for creating circular security counterexamples, which we will instantiate under a variety of different assumptions in the subsequent sections. At the center of our framework is an abstraction called a "cycle tester". Like an encryption scheme, a cycle tester must be able to encode a message in an IND-CPA secure manner. However, unlike an encryption scheme, the cycle tester need not support a decryption operation, instead it must support a testing operation which can detect the presence of an encryption cycle.

After formalizing this abstraction, we provide two results that use it. First, we show how our tester can be combined with any IND-CPA encryption scheme (of appropriate message length) to provide a full blown counterexample. Second, we extend this idea to show how to combine any tester with any IND-CCA encryption scheme to get an IND-CCA counterexample.

In addition to letting us focus on a narrower primitive for our counterexample, this separation avoids duplication of work and minimizes assumptions. In particular, we can design a single tester and then both the IND-CPA and IND-CCA counterexamples follow. Most prior works did not address IND-CCA counterexamples. While Cash et al. [17] did, their IND-CCA counterexample required the use of NIZKs, which is a stronger assumption than simply assuming the existence of IND-CCA encryption schemes as we do here. Our abstraction and transformation essentially show that designing IND-CCA counterexamples is no harder than designing IND-CPA ones.

We remark that Koppula et al. [24] have a IND-CPA counterexample with structure similar to our general transformation, however, no generic or IND-CCA theorems are proven.

Definition 4.1 (n-Cycle Tester) *A* cycle tester $\Gamma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Test})$ *for message space M and secret key space S is a tuple of algorithms specified as follows:*

- $\mathsf{Setup}(1^\lambda) \to \mathrm{PP}$. *The Setup algorithm takes as input the security parameter λ and outputs common public parameters* PP.
- $\mathsf{KeyGen}(\mathrm{PP}) \to (pk, sk)$. *The Key Generation algorithm takes as input the public parameters* PP *and outputs a public key pk and secret key $sk \in S$.*
- $\mathsf{Enc}(pk, m \in M) \to C$. *The Encryption algorithm takes as input a public key pk and a message $m \in M$ and outputs a ciphertext C.*
- $\mathsf{Test}(\mathbf{pk}, \mathbf{y}) \to \{0,1\}$. *On input $\mathbf{pk} = (pk_1, \ldots, pk_n)$ and $\mathbf{y} = (C_1, \ldots, C_n)$, the Testing algorithm outputs a bit in $\{0,1\}$.*

It also must possess the following properties. Let $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \cdot)$ be an encryption scheme formed from the first three algorithms of the tester with an empty decryption algorithm. Then, it must hold that:

1. *(IND-CPA security)* Π *is IND-CPA secure according to Definition 3.2.*
2. *(Testing Correctness) the Testing algorithm's advantage in distinguishing encryption cycles, denoted $\mathsf{Adv}^{n\text{-circ-cpa}}_{\Pi,\mathsf{Test}}(\lambda)$ from Definition 3.4, is non-negligible.*

We now prove two theorems.

Theorem 4.2 (CPA Counterexample from Cycle Testers) *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 n-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 n-circular insecure.*

Proof. Let $\Pi = (\mathsf{Setup}_1, \mathsf{KeyGen}_1, \mathsf{Enc}_1, \mathsf{Dec}_1)$ and $\Gamma = (\mathsf{Setup}_2, \mathsf{KeyGen}_2, \mathsf{Enc}_2, \mathsf{Test}_2)$. We construct an IND-CPA $\Pi' = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, together with its IND-CIRC-CPA[2] test algorithm Test, as follows.

$\mathsf{Setup}(1^\lambda)$**:** On input 1^λ, run $\mathrm{PP}_1 \leftarrow \mathsf{Setup}_1(1^\lambda)$ and $\mathrm{PP}_2 \leftarrow \mathsf{Setup}_2(1^\lambda)$. Output $\mathrm{PP} = (\mathrm{PP}_1, \mathrm{PP}_2)$.

$\mathsf{KeyGen}(\mathrm{PP})$**:** On input $\mathrm{PP} = (\mathrm{PP}_1, \mathrm{PP}_2)$, run $(pk_1, sk_1) \leftarrow \mathsf{KeyGen}_1(\mathrm{PP}_1)$ and $(pk_2, sk_2) \leftarrow \mathsf{KeyGen}_2(\mathrm{PP}_2)$. Output $pk = (pk_1, pk_2)$ and $sk = (sk_1, sk_2)$.

$\mathsf{Enc}(pk, m)$**:** On input $pk = (pk_1, pk_2)$ and $m = (m_1, m_2) \in M$, run $c_1 \leftarrow \mathsf{Enc}_1(pk_1, (m_1, m_2))$ and $c_2 \leftarrow \mathsf{Enc}_2(pk_2, m_2)$. Output $C = (c_1, c_2)$.

$\mathsf{Dec}(sk, C)$**:** On input $sk = (sk_1, sk_2)$ and $C = (c_1, c_2)$, output $\mathsf{Dec}_1(sk_1, c_1)$.

$\mathsf{Test}(\mathbf{pk}, \mathbf{y})$**:** On input $\mathbf{pk} = (pk_1, \ldots, pk_n)$ and $\mathbf{y} = (C_1, \ldots, C_n)$, parse $pk_i = (a_i, b_i)$ and $C_i = (c_i, d_i)$ and output the bit $\mathsf{Test}_2((b_1, \ldots, b_n), (d_1, \ldots, d_n))$.

The correctness of Test follows directly from that of Test_2. If $(\mathbf{pk}, \mathbf{y})$ contains an encryption cycle (or encryptions of zero, respectively), then so will $((b_1, \ldots, b_n), (d_1, \ldots, d_n))$, and thus by definition of the cycle tester, the test will distinguish between these cases with non-negligible advantage.

It remains to argue that Π' is an IND-CPA secure encryption scheme. This follows by a simple hybrid argument based on the fact that an encryption in Π' is a pair of encryptions from two different IND-CPA-secure schemes, Γ and Π. We omit this proof as it is a simplified version of the IND-CCA proof that we provide next.

□

Theorem 4.3 (CCA Counterexample from Cycle Testers) *Let k, ℓ be security parameters and $p(\cdot)$ be a polynomial. If there exists an IND-CCA-secure encryption scheme Π (with k-bit secret keys and $(p(\ell)+2k)$-bit messages) and an n-cycle tester Γ (with k-bit secret keys, k-bit messages, and $p(\ell)$-bit ciphertexts), then there exists an IND-CCA-secure encryption scheme Π' for $2k$-bit messages that is n-circular insecure.*

Proof. Let $\Pi = (\mathsf{Setup}_1, \mathsf{KeyGen}_1, \mathsf{Enc}_1, \mathsf{Dec}_1)$ and $\Gamma = (\mathsf{Setup}_2, \mathsf{KeyGen}_2, \mathsf{Enc}_2, \mathsf{Test}_2)$ with the length constraints above. We construct an IND-CCA $\Pi' = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, together with its IND-CIRC-CPA[2] test algorithm Test, as follows. We can no longer simply append the cycle-tester encryption to the regular encryption, because changes to the cycle-testing portion might be leveraged to obtain a decryption of a portion of the challenge ciphertext. Instead, we encrypt this cycle-testing portion using the regular CCA-secure scheme.

$\mathsf{Setup}(1^\lambda)$**:** On input 1^λ, run $\mathrm{PP}_1 \leftarrow \mathsf{Setup}_1(1^\lambda)$ and $\mathrm{PP}_2 \leftarrow \mathsf{Setup}_2(1^\lambda)$. Output $\mathrm{PP} = (\mathrm{PP}_1, \mathrm{PP}_2)$.

$\mathsf{KeyGen}(\mathrm{PP})$**:** On input $\mathrm{PP} = (\mathrm{PP}_1, \mathrm{PP}_2)$, run $(pk_1, sk_1) \leftarrow \mathsf{KeyGen}_1(\mathrm{PP}_1)$ and $(pk_2, sk_2) \leftarrow \mathsf{KeyGen}_2(\mathrm{PP}_2)$. Output $pk = (pk_1, pk_2)$ and $sk = (sk_1, sk_2)$.

$\mathsf{Enc}(pk, (m_a, m_b))$**:** On input $pk = (pk_1, pk_2)$ and message $(m_a, m_b) \in \{0,1\}^k \times \{0,1\}^k$, run $c_2 \leftarrow \mathsf{Enc}_2(pk_2, m_b)$ and $c_1 \leftarrow \mathsf{Enc}_1(pk_1, (m_a, m_b, c_2))$. Output $C = (c_1, c_2)$.

$\mathsf{Dec}(sk, C)$**:** On input $sk = (sk_1, sk_2)$ and $C = (c_1, c_2)$, run $\mathsf{Dec}_1(sk_1, c_1)$. If it does not return a message of the form $(m_a, m_b, m_c) \in \{0,1\}^k \times \{0,1\}^k \times \{0,1\}^{p(\lambda)}$ or if $m_c \neq c_2$, then output $\perp$ (invalid ciphertext). Otherwise, output the message $(m_a, m_b) \in \{0,1\}^k \times \{0,1\}^k$.

$\mathsf{Test}(\mathbf{pk}, \mathbf{y})$**:** On input $\mathbf{pk} = (pk_1, \ldots, pk_n)$ and $\mathbf{y} = (C_1, \ldots, C_n)$, parse $pk_i = (a_i, b_i)$ and $C_i = (c_i, d_i)$ and output the bit $\mathsf{Test}_2((b_1, \ldots, b_n), (d_1, \ldots, d_n))$. Same as before.

As before, the correctness of Test follows directly from that of Test_2. If $(\mathbf{pk}, \mathbf{y})$ contains an encryption cycle (or encryptions of zero, respectively), then so will $((b_1, \ldots, b_n), (d_1, \ldots, d_n))$, and thus by definition of the cycle tester, the test will distinguish between these cases with non-negligible advantage.

4.1 Proving IND-CCA Security via a Sequence of Games

It remains to argue that Π' is an IND-CCA secure encryption scheme. This proof is significantly more involved than the IND-CPA case. We prove this using a sequence of games from an encryption of a message M_0 to an encryption of M_1 (where these messages come from the IND-CCA game). The public and secret keys are always distributed as in the real scheme, but the structure of the challenge ciphertext changes in each hybrid. We underline these changes for the reader. Let the challenge messages be described as $M_0 = (m_{0,a}, m_{0,b})$ and $M_1 = (m_{1,a}, m_{1,b})$. Then the hybrids are as follows:

Game 1 This corresponds to the original security game $\mathsf{IND\text{-}CCA}(\Pi', \mathcal{A}, \lambda)$ in which the challenger interacts with adversary $\mathcal{A}$, except that the challenge ciphertext is always an encryption of message M_0.

1. Run $\mathsf{Setup}(1^\lambda)$ to produce PP and then $\mathsf{KeyGen}(\text{PP})$ to produce (pk, sk).
2. On decryption query C_i from $\mathcal{A}$, output $\mathsf{Dec}(sk, C_i)$.
3. Provide the challenge ciphertext as $C^* = (c_1^*, c_2^*)$, where $c_1^* = \mathsf{Enc}_1(pk_1, (m_{0,a},\ m_{0,b}, c_2^*))$ and $c_2^* = \mathsf{Enc}_2(pk_2, m_{0,b})$. This is a valid encryption of M_0.
4. On decryption query $C_i \neq C^*$ from $\mathcal{A}$, output $\mathsf{Dec}(sk, C_i)$.

Game 2 This is the same as Game 1, except that we change how the second decryption queries to reject *all* requests where the first portion of the query matches the first portion of the challenge.

1. Run $\mathsf{Setup}(1^\lambda)$ to produce PP and then $\mathsf{KeyGen}(\text{PP})$ to produce (pk, sk).
2. On decryption query C_i from $\mathcal{A}$, output $\mathsf{Dec}(sk, C_i)$.
3. Provide the challenge ciphertext as $C^* = (c_1^*, c_2^*)$, where $c_1^* = \mathsf{Enc}_1(pk_1, (m_{0,a},\ m_{0,b}, c_2^*))$ and $c_2^* = \mathsf{Enc}_2(pk_2, m_{0,b})$. This is a valid encryption of M_0.
4. On decryption query $C_i = (c_{i,1}, c_{i,2}) \neq C^*$ from $\mathcal{A}$, if $c_{i,1} = c_1^*$ output $\perp$, otherwise output $\mathsf{Dec}(sk, C_i)$.

Game 3 This is the same as Game 2, except that we now encrypt M_1 in the cycle tester portion and continue to encrypt M_0 in the regular encryption portion. We continue to reject all decryption queries where the regular encryption portion matches the challenge.

1. Run $\mathsf{Setup}(1^\lambda)$ to produce PP and then $\mathsf{KeyGen}(\text{PP})$ to produce (pk, sk).
2. On decryption query C_i from $\mathcal{A}$, output $\mathsf{Dec}(sk, C_i)$.
3. Provide the challenge ciphertext as $C^* = (c_1^*, c_2^*)$, where $c_1^* = \mathsf{Enc}_1(pk_1, (m_{0,a}, m_{0,b}, c_2^*))$ and $c_2^* = \mathsf{Enc}_2(pk_2, m_{1,b})$.
4. On decryption query $C_i = (c_{i,1}, c_{i,2}) \neq C^*$ from $\mathcal{A}$, if $c_{i,1} = c_1^*$ output $\perp$, otherwise output $\mathsf{Dec}(sk, C_i)$.

Game 4 This is the same as Game 3, except that now the entire challenge ciphertext is an encryption of M_1. As before, we continue to reject all decryption queries where the regular encryption portion matches the challenge.

1. Run $\mathsf{Setup}(1^\lambda)$ to produce PP and then $\mathsf{KeyGen}(\text{PP})$ to produce (pk, sk).
2. On decryption query C_i from $\mathcal{A}$, output $\mathsf{Dec}(sk, C_i)$.
3. Provide the challenge ciphertext as $C^* = (c_1^*, c_2^*)$, where $c_1^* = \mathsf{Enc}_1(pk_1, (m_{1,a},\ m_{1,b}, c_2^*))$ and $c_2^* = \mathsf{Enc}_2(pk_2, m_{1,b})$.
4. On decryption query $C_i = (c_{i,1}, c_{i,2}) \neq C^*$ from $\mathcal{A}$, if $c_{i,1} = c_1^*$ output $\perp$, otherwise output $\mathsf{Dec}(sk, C_i)$.

Game 5 This is the same as Game 4, except now all decryption queries are answered as normal. The challenge ciphertext always contains an encryption of M_1.

1. Run $\mathsf{Setup}(1^\lambda)$ to produce PP and then $\mathsf{KeyGen}(\text{PP})$ to produce (pk, sk).
2. On decryption query C_i from $\mathcal{A}$, output $\mathsf{Dec}(sk, C_i)$.
3. Provide the challenge ciphertext as $C^* = (c_1^*, c_2^*)$, where $c_1^* = \mathsf{Enc}_1(pk_1, (m_{1,a},\ m_{1,b}, c_2^*))$ and $c_2^* = \mathsf{Enc}_2(pk_2, m_{1,b})$.
4. On decryption query $C_i \neq C^*$ from $\mathcal{A}$, output $\mathsf{Dec}(sk, C_i)$.

4.2 Adversary's Probability of Outputting 1 in these Games

Let $\mathsf{Prob}^i_{\mathcal{A}}$ denote the probability that adversary $\mathcal{A}$ outputs a 1 in Game i. We will now show, by a series of steps, that for any adversary $\mathcal{A}$ the difference in its probability of outputting 1 between Game 1 (encryption of M_0) and Game 5 (encryption of M_1) is negligible. Thus, it cannot distinguish between these two games.

Claim 4.4 *For any adversary $\mathcal{A}$, $\mathsf{Prob}^2_{\mathcal{A}} = \mathsf{Prob}^1_{\mathcal{A}}$.*

Proof. These games are identical except that in Game 1 all decryption queries $C_i = C^*$ are rejected whereas in Game 2 all decryption queries $C_i = (c_{i,1}, c_{i,2})$ such that $c_{i,1} = c_1^*$ for $C^* = (c_1^*, c_2^*)$ are rejected. This results, however, in identical behavior on the decryption queries. Whenever $c_{i,1} \neq c_1^*$, both games answer the queries normally. Whenever $C_i = C^*$, neither game answers this illegal challenge query. On $c_{i,1} = c_1^*$ but $c_{i,2} \neq c_2^*$, Game 2 will output $\perp$. However, Game 1's response is also to reject this query with the message $\perp$ for being a non-valid ciphertext, since the decryption of c_1^* results in an intermediate tuple of the form $(m_{0,a}, m_{0,b}, c_2^*)$ and the decryption algorithm checks that $c_2^* = c_{i,2}$, which won't be true in this case. Thus, the adversary gets identical responses to its decryption queries (and everything else) in both games. Since the games are identical, from the adversary's viewpoint, it will output 1 with the same probability. □

Claim 4.5 *If Γ is an IND-CPA-secure n-cycle tester with security parameter λ, then for any adversary $\mathcal{A}$, $\mathsf{Prob}^3_{\mathcal{A}} - \mathsf{Prob}^2_{\mathcal{A}} \leq \mathsf{negl}(\lambda)$.*

Proof. We show that an attacker's probability of outputting 1 cannot be non-negligibly different in Games 2 and 3, because that would imply an attack on the IND-CPA security of the cycle tester. More formally, suppose there exists an adversary $\mathcal{A}$ such that $\mathsf{Prob}^3_{\mathcal{A}} - \mathsf{Prob}^2_{\mathcal{A}} = \epsilon$. Then we can construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to show that Γ is not an IND-CPA-secure n-cycle tester. $\mathcal{B}$ works as follows:

1. $\mathcal{B}$ runs $\mathsf{Setup}_1(1^\lambda) \to \text{PP}_1$ and $\mathsf{KeyGen}_1(\text{PP}_1) \to (pk_1, sk_1)$.
2. $\mathcal{B}$ obtains the public key pk_2 from the IND-CPA encryption challenger.
3. $\mathcal{B}$ sends $pk = (pk_1, pk_2)$ to $\mathcal{A}$.
4. $\mathcal{A}$ returns two messages $M_0 = (m_{0,a}, m_{0,b})$ and $M_1 = (m_{1,a}, m_{1,b})$.
5. $\mathcal{B}$ sends $(m_{0,b}, m_{1,b})$ to the cycle tester encryption challenger and obtains the challenge c_2^*.
6. $\mathcal{B}$ forms the challenge ciphertext by computing $c_1^* = \mathsf{Enc}_1(pk_1, (m_{0,a}, m_{0,b}, c_2^*))$ and sending $C^* = (c_1^*, c_2^*)$ to $\mathcal{A}$.
7. Eventually, $\mathcal{A}$ returns a bit $\hat{b}$ and $\mathcal{B}$ outputs $\hat{b}$ to its challenger.

In the above, $\mathcal{B}$ perfectly simulates Game 2 for adversary $\mathcal{A}$ if the challenge ciphertext c_2^* contains an encryption of $m_{0,b}$ and, in the other case, $\mathcal{B}$ perfectly simulates Game 3 for adversary $\mathcal{A}$ when the challenge ciphertext c_2^* contains an encryption of $m_{1,b}$. Moreover, $\mathcal{B}$ succeeds if and only if $\mathcal{A}$ succeeds. Thus, if $\mathsf{Prob}^3_{\mathcal{A}} - \mathsf{Prob}^2_{\mathcal{A}} = \epsilon$, then we have $\Pr[\mathcal{B} \text{ is correct}] = \frac{1}{2}\Pr[\mathcal{B} \text{ is correct} \mid \text{IND-CPA challenger chose 0}] + \frac{1}{2}\Pr[\mathcal{B} \text{ is correct} \mid \text{IND-CPA challenger chose 1}] = \frac{1}{2}\Pr[\mathcal{A} \text{ is correct} \mid \text{Game 2}] + \frac{1}{2}\Pr[\mathcal{A} \text{ is correct} \mid \text{Game 3}] = \frac{1}{2}(1 - \mathsf{Prob}^2_{\mathcal{A}}) + \frac{1}{2}(\mathsf{Prob}^3_{\mathcal{A}}) = \frac{1}{2}(1 - \mathsf{Prob}^2_{\mathcal{A}}) + \frac{1}{2}(\mathsf{Prob}^2_{\mathcal{A}} + \epsilon) = \frac{1}{2} + \frac{\epsilon}{2}$. Since we assumed the cycle tester was IND-CPA secure, it must hold that $\epsilon \leq \mathsf{negl}(\lambda)$. □

Claim 4.6 *If Π is an IND-CCA-secure encryption scheme with security parameter λ, then for any adversary $\mathcal{A}$, $\mathsf{Prob}^4_{\mathcal{A}} - \mathsf{Prob}^3_{\mathcal{A}} \leq \mathsf{negl}(\lambda)$.*

Proof. Suppose there exists an adversary $\mathcal{A}$ such that $\mathsf{Prob}^4_{\mathcal{A}} - \mathsf{Prob}^3_{\mathcal{A}} = \epsilon$. Then we can construct an adversary $\mathcal{B}$ that uses $\mathcal{A}$ to show that Π is not an IND-CCA-secure encryption scheme. $\mathcal{B}$ works as follows:

1. $\mathcal{B}$ obtains the public key pk_1 from the IND-CCA encryption challenger.
2. $\mathcal{B}$ runs $\mathsf{Setup}_2(1^\lambda) \to \mathrm{PP}_2$ and $\mathsf{KeyGen}_2(\mathrm{PP}_2) \to (pk_2, sk_2)$.
3. $\mathcal{B}$ sends $pk = (pk_1, pk_2)$ to $\mathcal{A}$.
4. On receiving a decryption query for ciphertext $C_i = (c_{i,1}, c_{i,2})$ from $\mathcal{A}$, $\mathcal{B}$ sends $c_{i,1}$ to its IND-CCA encryption challenger to obtain a message M. $\mathcal{B}$ returns M to $\mathcal{A}$.
5. $\mathcal{A}$ returns two messages $M_0 = (m_{0,a}, m_{0,b})$ and $M_1 = (m_{1,a}, m_{1,b})$.
6. $\mathcal{B}$ computes $c_2^* = \mathsf{Enc}_2(pk_2, m_{1,b})$ and sends $M_0' = (M_0, c_2^*)$ and $M_1' = (M_1, c_2^*)$ to the IND-CCA challenger and obtains the challenge c_1^*.
7. $\mathcal{B}$ sends the challenge ciphertext $C^* = (c_1^*, c_2^*)$ to $\mathcal{A}$.
8. On receiving a decryption query for ciphertext $C_i = (c_{i,1}, c_{i,2})$ where $c_{i,1} \neq c_1^*$ from $\mathcal{A}$, $\mathcal{B}$ sends $c_{i,1}$ to its IND-CCA encryption challenger to obtain a message M. $\mathcal{B}$ returns M to $\mathcal{A}$
9. Eventually, $\mathcal{A}$ returns a bit $\hat{b}$ and $\mathcal{B}$ outputs $\hat{b}$ to its challenger.

In the above, $\mathcal{B}$ perfectly simulates Game 3 for adversary $\mathcal{A}$ if the challenge ciphertext c_1^* contains an encryption of M_0' and, in the other case, $\mathcal{B}$ perfectly simulates Game 4 for adversary $\mathcal{A}$ when the challenge ciphertext c_1^* contains an encryption of M_1'. Moreover, $\mathcal{B}$ succeeds if and only if $\mathcal{A}$ succeeds. Thus, if $\mathsf{Prob}^4_{\mathcal{A}} - \mathsf{Prob}^3_{\mathcal{A}} = \epsilon$, then $\mathcal{B}$'s probability of success in the IND-CCA security game is $\Pr[\mathcal{B} \text{ is correct}] = \frac{1}{2}\Pr[\mathcal{B} \text{ is correct} \mid \text{IND-CCA challenger chose 0}] + \frac{1}{2}\Pr[\mathcal{B} \text{ is correct} \mid \text{IND-CCA challenger chose 1}] = \frac{1}{2}\Pr[\mathcal{A} \text{ is correct} \mid \text{Game 3}] + \frac{1}{2}\Pr[\mathcal{A} \text{ is correct} \mid \text{Game 4}] = \frac{1}{2}(1 - \mathsf{Prob}^3_{\mathcal{A}}) + \frac{1}{2}(\mathsf{Prob}^4_{\mathcal{A}}) = \frac{1}{2}(1 - \mathsf{Prob}^3_{\mathcal{A}}) + \frac{1}{2}(\mathsf{Prob}^3_{\mathcal{A}} + \epsilon) = \frac{1}{2} + \frac{\epsilon}{2}$. Since we assumed that Π was IND-CCA secure, it must hold that $\epsilon \leq \mathsf{negl}(\lambda)$. □

Claim 4.7 *For any adversary $\mathcal{A}$, $\mathsf{Prob}^5_{\mathcal{A}} = \mathsf{Prob}^4_{\mathcal{A}}$.*

Proof. These games are identical except that in Game 4 all decryption queries $C_i = (c_{i,1}, c_{i,2})$ such that $c_{i,1} = c_1^*$ for $C^* = (c_1^*, c_2^*)$ are rejected in Game 5 whereas all decryption queries $C_i = C^*$ are rejected. This results, however, in identical behavior on the decryption queries. This case is the mirror image of the argument in the proof of Claim 4.4. □

Conclusion of the Proof of Theorem 4.3 Given the above claims, we can conclude that if Γ is an IND-CPA-secure n-cycle tester and Π is an IND-CCA-secure encryption scheme (with the appropriate length constraints), then for any adversary $\mathcal{A}$, it holds that $\mathsf{Prob}^5_{\mathcal{A}} - \mathsf{Prob}^1_{\mathcal{A}}$ is negligible, implying that Π' is an IND-CCA-secure encryption scheme.

□

5 Warm up: 2-Cycle Tester from the SXDH Assumption

We start with a simple 2-cycle tester set in Type-III pairing groups where the SXDH assumption holds. This is derived from a prior SXDH-based counterexample of Cash, Green and Hohenberger [17] and is also similar to the example of Acar et al. [1].

Our construction is described below. In it we slightly abuse notation and let our message space $\mathbb{Z}_p^*$ be defined by our Setup algorithm. In Section 6 we show how to remedy this by having a message space be $\{0,1\}^\lambda$ (that only depends on the security parameter λ) and having the Setup algorithm define a pseudorandom generator which maps from $\{0,1\}^\lambda$ to the algebraic message space needed for the construction.

$\mathsf{Setup}(1^\lambda) \to \mathrm{PP}$. Recall we assume a setting where all parties implicitly use shared public parameters. Run $\mathcal{G}(1^\lambda)$ to generate a Type-III pairing description $\mathrm{PP} = (p, g, h, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, such that p is a prime in $\Theta(2^\lambda)$, $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are groups of order p where g generates $\mathbb{G}_1$, h generates $\mathbb{G}_2$ and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an admissible pairing. The message space is $\mathbb{Z}_p^*$.

$\mathsf{KeyGen}(\mathrm{PP}) \to (pk, sk)$. Flip a coin $\beta \in \{0,1\}$ and choose random $s \in \mathbb{Z}_p^*$. Set the public key as:

$$pk = \begin{cases} (0, g^s) \in \{0,1\} \times \mathbb{G}_1 & \text{if } \beta = 0; \\ (1, h^s) \in \{0,1\} \times \mathbb{G}_2 & \text{if } \beta = 1. \end{cases}$$

The secret key sk is s, making the secret key space $\mathbb{Z}_p^*$ (which is the same as the message space).

$\mathsf{Enc}(pk, m \in \mathbb{Z}_p^*) \to C$. To encrypt, parse $pk = (\beta, K)$ and choose a random $t \in \mathbb{Z}_p^*$. Generate the ciphertext $C = (C_1, C_2)$ as:

$$C = \begin{cases} (C_1 = K^{tm} = g^{stm},\ \ C_2 = g^t) \in \mathbb{G}_1^2 & \text{if } \beta = 0; \\ (C_1 = K^{tm} = h^{stm},\ \ C_2 = h^t) \in \mathbb{G}_2^2 & \text{if } \beta = 1. \end{cases}$$

$\mathsf{Test}(\mathbf{pk}, \mathbf{y}) \to \{0,1\}$. Since we are testing for 2-cycles, parse $\mathbf{pk} = (pk = (\beta, K), pk' = (\beta', K'))$ and $\mathbf{y} = (C = (C_1, C_2), C' = (C_1', C_2'))$. If $\beta = \beta'$, output a random bit. Otherwise, assume without loss of generality that $\beta = 0$ and $\beta' = 1$. Then the test algorithm checks if

$$e(C_1, C_2') \stackrel{?}{=} e(C_2, C_1').$$

If this check holds, then output 1 (guess cycle); otherwise output 0 (guess zeros).

Analysis of Test Algorithm In the $\mathsf{IND\text{-}CIRC\text{-}CPA}^2$ game, two public keys $pk = (\beta, K), pk' = (\beta', K)$ are freshly generated by the KeyGen algorithm based on a shared common parameters freshly generated by the Setup algorithm. Since β and β' are independently chosen bits, the probability that $\beta = \beta'$ is $1/2$. In this case, the Test algorithm will output a random bit (i.e., a random guess) and thus will be successful exactly $1/2$ the time.

Otherwise, we assume without loss of generality that $\beta = 0$ and $\beta' = 1$ (if not, swap the two key/ciphertext pairs). In this game, the challenge ciphertexts (C, C') are formed using the Enc algorithm. By plugging these values into the Test equation above, we have:

$$\begin{aligned} e(g^{stm}, h^{t'}) &= e(g^t, h^{s't'm'}) \\ e(g,h)^{stmt'} &= e(g,h)^{s't'm't} \\ e(g,h)^{sm} &= e(g,h)^{s'm'} \end{aligned}$$

Recall that $t, t' \in \mathbb{Z}_p^*$, so we can divide them out of the exponent. From the last equation, it is clear that if there is a cycle (meaning $m = s'$ and $m' = s$), then both sides equal $e(g,h)^{ss'}$ and the equation holds. If we

have an encryption of zero (recall this is encoded as some element $z \in \mathbb{Z}_p^*$, so $z \neq 0$), then the left-hand-side is $e(g,h)^{sz}$ and the right-hand-side is $e(g,h)^{s'z}$, which will only be equal if $s = s'$. Since these secret keys were randomly chosen from $\mathbb{Z}_p^*$, the chance they are the same is $1/(p-1)$, which is negligible in λ. Thus, the success of the Test algorithm is (1/2)(1/2) + (1/2)((1/2) 1 + (1/2) (1-1/(p-1)) = 3/4 - $\mathsf{negl}(\lambda)$, which is more than sufficient probability to win the $\mathsf{IND\text{-}CIRC\text{-}CPA}^2$ game.

IND-CPA Security of the Tester

Theorem 5.1 *The above encryption scheme* $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \cdot)$ *(where the decryption algorithm is ignored) is* $\mathsf{IND\text{-}CPA}$*-secure under the SXDH Assumption in* $\mathbb{G}_1, \mathbb{G}_2$.

Proof. Suppose that there exists a PPT adversary $\mathcal{A}$ that has a non-negligible advantage ϵ in the $\mathsf{IND\text{-}CPA}$ security game. We can then construct a PPT algorithm $\mathcal{B}$ that solves the SXDH problem with advantage $\frac{1}{2} + \frac{\epsilon}{2}$. $\mathcal{B}$ works as follows:

1. Select a random bit $\beta \in \{0,1\}$. If $\beta = 0$, we will obtain a DDH instance $(g, g^a, g^b, T_d) \in \mathbb{G}_1^4$. If $\beta = 1$, we will obtain the same instance in $\mathbb{G}_2$. We proceed assuming $\beta = 0$; the case of $\beta = 1$ is analogous.
2. Set the public key pk as
$$pk = (\beta, K = g^s = g^a)$$
(implicitly setting $s = a$) and output it to $\mathcal{A}$. Note that $\mathcal{B}$ does not know the secret key.
3. Run $\mathcal{A}(pk)$ to produce a tuple (m_0, m_1).
4. Select a random bit $\gamma \in \{0,1\}$.
5. Set the challenge ciphertext as $C^* = (C_1 = T_d^{m_\gamma}, C_2 = g^b) \in \mathbb{G}_1^2$.
6. Run $\mathcal{A}(C^*)$ to get a bit γ'. If $\gamma = \gamma'$, then output 0 (guess a DDH instance). Otherwise, output 1 (guess T_d was random).

If $T_d = g^{ab}$, then $C_1 = T_d^{m_\gamma} = g^{abm_\gamma} = g^{stm_\gamma}$ and the ciphertext is well-formed according to the original encryption algorithm. $\mathcal{B}$ wins with probability $\frac{1}{2} + \epsilon$ in this case, since $\mathcal{A}$ does. Otherwise, C_1 is a random group element, which means that $\mathcal{A}$ guesses correctly with probability $\frac{1}{2}$. In this case, $\mathcal{B}$ wins with probability $\frac{1}{2}$ as well. $\mathcal{B}$ then has an overall success probability of $\frac{1}{2}\left(\frac{1}{2} + \epsilon\right) + \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2} + \frac{\epsilon}{2}$, which gives a non-negligible advantage of $\frac{\epsilon}{2}$.

□

6 A 2-Cycle Tester from the k-DLIN Assumption

We now present a 2-cycle tester from the decision k-Linear assumption in pairing groups for any constant k (where this assumption is believed to hold for $k \geq 2$ in this bilinear setting and the assumption grows weaker as k increases). We will use a message space of $\{0,1\}^\lambda$. In our exposition we will use boldface to denote a matrix such as $\mathbf{M}$. We also use $g^{\mathbf{M}}$ as shorthand to denote the group elements corresponding to the raising g to each individual element of $\mathbf{M}$.

$\mathsf{Setup}(1^\lambda) \to \mathrm{PP}$**.** The setup algorithm first runs $\mathcal{G}(1^\lambda)$ to generate a (Type-1) group $\mathbb{G}$ of prime order p with generator g. Next it defines a pseudorandom generator $\mathrm{PRG} : \{0,1\}^\lambda \to \mathbb{Z}_p^{k \times k}$, which maps strings from $\{0,1\}^\lambda$ to *invertible* $k \times k$ matrices over $\mathbb{Z}_p$. Finally, it chooses a random invertible matrix $\mathbf{A} \in \mathbb{Z}_p^{k \times k}$ and computes $g^{\mathbf{A}}$. The public parameters, PP consist of the group description $\mathbb{G}$, the description of PRG and $g^{\mathbf{A}}$.

$\mathsf{KeyGen}(\mathrm{PP}) \to (pk, sk)$**.** The key generation algorithm first chooses random $w \in \{0,1\}^\lambda$. The secret key $sk = w$. Next, it computes $\mathrm{PRG}(w) \to \mathbf{W} \in \mathbb{Z}_p^{k \times k}$ and chooses a bit $\beta \in \{0,1\}$. Finally, in addition to implicitly including PP, it defines the public key as

$$pk = \begin{cases} (0, K = g^{\mathbf{AW}}) \in \{0,1\} \times \mathbb{G}^{k \times k} & \text{if } \beta = 0; \\ (1, K = g^{\mathbf{WA}}) \in \{0,1\} \times \mathbb{G}^{k \times k} & \text{if } \beta = 1. \end{cases}$$

$\mathsf{Enc}(pk = (\beta, K), m \in \{0,1\}^\lambda) \to ct$**.** The encryption algorithm first computes computes $\mathrm{PRG}(m) \to \mathbf{M} \in \mathbb{Z}_p^{k \times k}$ and then computes $\mathbf{M}^{-1}$. Note that since PRG maps to invertible matrices, $\mathbf{M}$ will have an inverse.

If the type bit $\beta = 0$ the key $K = g^{\mathbf{AW}}$ for some $\mathbf{W}$. The algorithm chooses $\mathbf{r}$ as a random row vector of length k in $\mathbb{Z}_p$ (i.e. a random matrix of dimension $1 \times k$). The ciphertext is computed and output as

$$C_1 = g^{\mathbf{rAW}}, \quad C_2 = g^{\mathbf{rAM}^{-1}}.$$

Thus, the ciphertext will consist of two row vectors in the exponent. We observe all terms are computable from the public keys and public parameters.

If the type bit $\beta = 1$ the key $K = g^{\mathbf{WA}}$ for some $\mathbf{W}$. The algorithm chooses $\mathbf{r}$ as a random column vector of length k in $\mathbb{Z}_p$ (i.e. a random matrix of dimension $k \times 1$). The ciphertext ct is computed and output as

$$C_1 = g^{\mathbf{WAr}}, \quad C_2 = g^{\mathbf{M}^{-1}\mathbf{Ar}}.$$

$\mathsf{Test}(\mathbf{pk}, \mathbf{y}) \to \{0,1\}$**.** Since we are testing for 2-cycles, parse $\mathbf{y} = (C = (C_1, C_2)$, $C' = (C_1', C_2'))$. If the key types are identical i.e. $\beta = \beta'$ then just output a random bit as a guess.

Otherwise, presume that $\beta = 0, \beta' = 1$ (if it is the other way around just flip the order). Then compute $e(C_1, C_2') \stackrel{?}{=} e(C_1', C_2)$ and output the result. Note here we overload notation so that the pairing operator e is over a matrix of group elements and means matrix multiplication in the exponent. (Or in this case a dot product in the exponent.)

Analysis of Test Algorithm We analyze the correctness of the test algorithm. Let's consider two secret keys w, w' where $\mathrm{PRG}(w) = \mathbf{W}$ and $\mathrm{PRG}(w') = \mathbf{W}'$. Again, presume that $\beta = 0, \beta' = 1$. The corresponding public keys will be $pk = g^{\mathbf{AW}}$ and $pk = g^{\mathbf{W}'\mathbf{A}}$. Now consider an encryption of m under pk and m' under pk' where $\mathrm{PRG}(m) = \mathbf{M}$ and $\mathrm{PRG}(M') = \mathbf{M}'$. Let $\mathbf{r}$ and $\mathbf{r}'$ be the respective randomness used for each encryption.

The test equations outputs 1 iff $e(C_1, C_2') \stackrel{?}{=} e(C_1', C_2)$ this is equivalent to testing

$$\mathbf{rAWM}'^{-1}\mathbf{Ar}' \stackrel{?}{=} \mathbf{rAM}^{-1}\mathbf{W}'\mathbf{Ar}'. \tag{1}$$

Let's first consider the case where we have an encryption of a cycle. This means that $m' = w$ and $m = w'$ so we have that $\mathbf{M}'^{-1} = \mathbf{W}^{-1}$ and $\mathbf{M}^{-1} = \mathbf{W}'^{-1}$. Substituting these in we see that

$$\begin{aligned} \mathbf{rAWM}'^{-1}\mathbf{Ar}' &\stackrel{?}{=} \mathbf{rAM}^{-1}\mathbf{W}'\mathbf{Ar}' \\ \mathbf{rAIAr}' &\stackrel{?}{=} \mathbf{rAIAr}' \\ \mathbf{rA}^2\mathbf{r}' &\stackrel{?}{=} \mathbf{rA}^2\mathbf{r}'. \end{aligned}$$

Thus, on a cycle the test will output 1.

We now turn to the case of showing that an encryption of 0's will output 0 (when the keys have different β types) with all but negligible probability.

First, we first let $\mathbf{Z} = \mathrm{PRG}(0^\lambda)^{-1}$ which is the matrix used to encrypt the all 0's string. Second, we consider the probability of the tester outputting 1, when $\mathbf{W}$ and $\mathbf{W}'$ are chosen uniformly at random (and independently from $\mathbf{Z}$) from the set of full rank matrices, as opposed to being the output of a pseudorandom

generator. If there, was more than a negligible difference of the test in outputting 1 in these two cases, it would lead to an attack on the security of the pseudorandom generator.

We can now observe that the matrices $\mathbf{X} = \mathbf{AWZA}$ and $\mathbf{X}' = \mathbf{AZW}'\mathbf{A}$ are distributed independently and uniformly random from full rank matrices. Note we substituted $\mathbf{Z}$ for both $\mathbf{M}'^{-1}$ and $\mathbf{M}^{-1}$ in Equation 1. Then $\mathbf{u} = \mathbf{rX}$ and $\mathbf{u}' = \mathbf{rX}'$ are independently distributed as uniformly at random row vectors of length k. Finally, it follows that the probability that

$$\mathbf{ur}' \stackrel{?}{=} \mathbf{u}'\mathbf{r}'$$

is negligible in the security parameter. Thus, with probability negligibly close to 1 the test algorithm will output 0 when given an encryption of all 0's.

IND-CPA Security of the Tester

Theorem 6.1 *The above encryption scheme* $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Test})$ *(where the decryption algorithm is ignored) is* IND-CPA*-secure under the k-Linear Assumption in* $\mathbb{G}$.

Proof.

We prove security for the $\beta = 0$ case. The $\beta = 1$ case will follow analogously. We first define a sequence of games.

Game 1 This game is the IND-CPA security game for our construction

1. The challenger runs $\mathsf{Setup}(1^\lambda) \to \text{PP} = (\mathbb{G}, \text{PRG}, g^{\mathbf{A}})$.
2. Recall, we consider a $\beta = 0$ key. The challenger runs $\text{PRG}(w) \to \mathbf{W} \in \mathbb{Z}_p^{k\times k}$. Let $\mathbf{V} = \mathbf{AW}$. The key is computed as $K = g^{\mathbf{V}}$. The challenger gives the parameters, PP and key $pk^* = (0, K)$ to the attacker.
3. $\mathcal{A}$ The attacker submits two messages m_0, m_1 to the challenger.
4. The challenger flips a coin $b \in \{0, 1\}$. Let $\mathbf{M}_b = \text{PRG}(m_b) \in \mathbb{Z}_p^{k\times k}$. The challenger chooses $\mathbf{r}$ as a random row vector of length k in $\mathbb{Z}_p$. The challenge ciphertext is computed and output as

$$C_1^* = g^{\mathbf{rV}}, \quad C_2^* = g^{\mathbf{rAM}_b^{-1}}.$$

5. The attacker receives the challenge ciphertext. It then outputs a guess b' and wins if $b' = b$.

Game 2

2. Choose $\mathbf{V}$ as a random matrix of rank k in $\mathbb{Z}_p^{k\times k}$. The key is computed as $K = g^{\mathbf{V}}$. The challenger gives the parameters, PP and key $pk^* = (0, K)$ to the attacker.

Game 3

4. The challenger flips a coin $b \in \{0, 1\}$. Let $\mathbf{M}_b = \text{PRG}(m_b) \in \mathbb{Z}_p^{k\times k}$. The challenger chooses $\mathbf{r}, \mathbf{u}$ as a random row vectors of length k in $\mathbb{Z}_p$. The challenge ciphertext is computed and output as $C_1^* = g^{\mathbf{rV}}$, $C_2^* = g^{\mathbf{uAM}_b^{-1}}$.

Claim 6.2 *If the pseudorandom generator* PRG *is secure, then any PPT attacker will have at most a negligible difference in advantage between* **Game 1** *and* **Game 2**.

Proof. Suppose there exists an attacker $\mathcal{A}$. We build a reduction algorithm $\mathcal{B}$. $\mathcal{B}$ will first run **Game 1** with the exception that it will received $\mathbf{W} \in \mathbb{Z}_p^{k \times k}$ from the pseudorandom generator challenger. It then computes $\mathbf{V} = \mathbf{AW}$ and the key as. It then runs the rest of the experiment as in **Game 1**. If the attacker is successful in guessing the challenge bit b, $\mathcal{B}$ outputs 1 to indicate pseudorandom; else it outputs 0 to indicate random.

When $\mathbf{W}$ is output from the PRG the reduction exports the view of **Game 1** to $\mathcal{A}$, thus the probability of $\mathcal{B}$ outputting 1 will be the probability A wins in **Game 1**. If $\mathbf{W}$ is selected as a random matrix of rank k in $\mathbb{Z}_p^{k \times k}$, then the reduction exports the view of **Game 2**. This follows since the multiplication of a random full rank matrix by another full rank matrix is distributed the same as a random full rank matrix $\mathbf{V}$. Thus the probability of $\mathcal{B}$ outputting 1 (in this case) will be the probability A wins in **Game 2**. Therefore if any PPT algorithm $\mathcal{A}$ has a non-neglgibile difference in advantage between **Game 1** and **Game 2**, there will exist a PPT attack algorithm $\mathcal{B}$.

□

Claim 6.3 *If the decision k-linear assumption holds* PRG*, then any PPT attacker will have at most a negligible difference in advantage between* **Game 2** *and* **Game 3**.

Proof.

We prove this theorem relying on an instance of the decision matrix linear assumption introduced by Naor and Segev [28]. In our particular instance the challenger with give out a dimension $(k+1) \times 2k$ matrix $g^{\mathbf{T}}$ where $\mathbf{T}$ is either chosen as a random rank k matrix or a random rank $k+1$ matrix.

Our reduction algorithm will receive the group description $\mathbb{G}$ and the challenge $g^{\mathbf{T}}$. If $\mathbf{T}$ was chosen as a random rank k matrix it is statistically close to a matrix where the the top k rows are two $k \times k$ matrices $\mathbf{V}$ and $\mathbf{A}$ concatenated together. While the bottom row of $\mathbf{T}$ consists of a linear combination of the previous k rows of $\mathbf{T}$ and can be written as $\mathbf{rT} = \mathbf{rV}|\mathbf{rA}$ for random length k row vector $\mathbf{r}$. Whereas if $\mathbf{T}$ was chosen as a random rank $k+1$ matrix then the last row would be (statistically close) to a random row vector of length $2k$ and could be written as $\mathbf{rV}|\mathbf{uA}$for random length k row vectors $\mathbf{r}$ and $\mathbf{u}$.

Our reduction algorithm $\mathcal{B}$ receives the challenge $g^{\mathbf{T}}$ and proceeds to run the security game. It uses the top k rows of the matrix to simulate giving out the parameters $g^{\mathbf{A}}$ and $K = g^{\mathbf{V}}$ as described above. Then it uses the left half of the bottom row of the matrix to output $C_1^* = g^{\mathbf{rV}}$ and the right half of the bottom row to output $C_2^* = g^{\mathbf{rAM}_b^{-1}}$ or $C_2^* = g^{\mathbf{uAM}_b^{-1}}$ (depending on what the challenge was). We see that if the challenge was a rank k matrix, then we are in **Game 2**. If the challenge was a rank $k+1$ matrix, then $\mathcal{B}$ exports **Game 3**. Therefore any attacker than had a non-negligible difference of advantage between these games would break the decision matrix k linear assumption with non-negligible advantage. Since Naor and Segev showed this to be equivalent to the decision k-linear assumption our claim holds.

□

Claim 6.4 *The advantage of any attacker in* **Game 3** *is 0.*

Proof. Since $\mathbf{A}$ and $\mathbf{M}_b^{-1}$ are full rank k the matrix $\mathbf{AM}_b^{-1}$ is also of rank k. Since the row vector $\mathbf{u}$ is chosen randomly and does not appear elsewhere the row vector $\mathbf{uAM}_b^{-1}$ has the same distribution as a uniformly chosen row vector $\tilde{\mathbf{u}}$. In particular, the distribution is independent of the bit b and thus the ciphertext is independent of b.

□

Our theorem follows from the fact that any PPT algorithm will have 0 advantage in the last game and its advantage in the first game must be negligibly close to this.

□

7 A 2-Cycle Tester from Learning with Errors

We now present a 2-Cycle Tester whose IND-CPA security follows from the Learning with Errors Assumption. We note that our construction is similar to multi-bit Regev encryption.

7.1 Construction

$\mathsf{Setup}(1^n) \to \mathrm{PP}$. The setup algorithm chooses $m, q, \ell, \sigma, r, \alpha$. These parameters are chosen to satisfy the following constraints: $m \geq 2n \log q$, $\sigma \geq L\omega(\sqrt{\log m})$, $q \geq 5\sigma(m+1)$, $\ell > (n+m+1)\log q + \omega(\log(n+m))$, $r := \sigma\ell$, $\alpha \leq 1/(r\sqrt{m+1} \cdot \omega(\sqrt{\log n}))$, and $q > 2$ is prime. Here, L is defined as follows. We let z denote the number of uniform random bits employed by TrapGen to generate a matrix B in $\mathbb{Z}_q^{n \times m}$ along with a trapdoor basis T_B. L is a bound such that $||T_B||_{GS} \leq L$ with overwhelming probability. (We note that this range of parameters allows us to set α so that n/α is polynomial, and LWE is believed to be hard in this parameter regime.) The public parameters are $\mathrm{PP} = (m, q, \ell, \sigma, r, \alpha, z)$.

$\mathsf{KeyGen}(\mathrm{PP}) \to (pk, sk)$. The key generation algorithm chooses a uniformly random secret key sk in $\{0,1\}^z$ and runs $\mathrm{TrapGen}(sk)$ to produce a matrix $B \in \mathbb{Z}_q^{n \times m}$ and a corresponding trapdoor basis T_B. It then chooses independent and uniformly random vectors $c_1, \ldots, c_\ell \in \mathbb{Z}_q^n$ and noise vectors $\gamma_1, \ldots, \gamma_\ell$ from χ^m, where χ is distributed as $\lfloor q\Psi_\alpha \rfloor \mod q$, where Ψ_α is a distribution on $\mathbb{T}$ of a normal variable with mean 0 and standard deviation $\alpha/\sqrt{2\pi}$ reduced modulo 1. (We think of these vectors as row vectors.) In addition to implicitly including the PP, it sets

$$pk = \{c_1, \ldots, c_\ell, y_1 := c_1 B + \gamma_1, \ldots, y_\ell := c_\ell B + \gamma_\ell\}.$$

$\mathsf{Enc}(pk, m \in \{0,1\}^z) \to ct$. The encryption algorithm runs $\mathrm{TrapGen}(m)$ to produce a matrix $Z \in \mathbb{Z}_q^{n \times m}$ and a corresponding trapdoor basis T_Z. It chooses random signs $r_1, \ldots, r_\ell \in \{-1, 1\}$ and computes $s := \sum_{i=1}^{\ell} r_i c_i$. It then uses T_Z to sample a short (column) vector v such that $Zv = s^t$, by calling the algorithm SampleD. It computes

$$C = \sum_{i=1}^{\ell} r_i y_i,$$

and sets the ciphertext as $ct = (C, v)$.

$\mathsf{Test}((pk_0, pk_1), ((C_0, v_0), (C_1, v_1))) \to \{0,1\}$. The cycle test algorithm compares $C_0 v_1$ to $C_1 v_0$ and checks if there are close modulo q (if their distance is $\leq 2q/5$). If so, it outputs 1. If not, it outputs 0.

Analysis of Test Algorithm We let B_0, Z_0, s_0 be the B, Z and s values corresponding to ciphertext (C_0, v_0) and B_1, Z_1, s_1 be the analogous values for (C_1, v_1). When there is a cycle, we then have $Z_0 = B_1$ and $Z_1 = B_0$. We then have $B_0 v_1 = s_1^t$ and $B_1 v_0 = s_0^t$. Noting that $C_0 = s_0 B_0 + \psi_0$ for some small vector ψ_0, we see that

$$C_0 v_1 = s_0 B_0 v_1 + \psi_0 v_1 = s_0 s_1^t + \psi_0 v_1.$$

Similarly, $C_1 = s_1 B_1 + \psi_1$ for some small vector ψ_1, so we have that

$$C_1 v_0 = s_1 B_1 v_0 + \psi_1 v_0 = s_1 s_0^t + \psi_1 v_0.$$

We consider the size of $\psi_0 v_1 - \psi_1 v_0$ modulo q. First, $|\psi_0 v_1|$ is at most ℓ times the maximal size of $|\gamma_j v_1|$. Using the same analysis as in the proof of Lemma 8.2 of [20], each of these is $\leq \frac{q}{5\ell}$ with high probability. Thus, $|\psi_0 v_1 - \psi_1 v_0| \leq \frac{2q}{5}$ with high probability.

Since all of v_0, v_1, ψ_0, ψ_1 are short, this will cause these values to be close modulo q, so the cycle test will output 1 with high probability.

When there is no cycle, the matrices B_0 and B_1 are (statistically close) to independent, uniformly random matrices. Thus the probability that $s_0 B_0 v_1$ and $s_1 B_1 v_0$ will be within $\frac{2}{5}q$ modulo q is negligibly close to $\frac{2}{5}$. Thus the cycle test wins the distinguishing game with probability negligibly close to $\frac{1}{2} + \frac{1}{2} \cdot \frac{3}{5} = \frac{4}{5}$.

7.2 IND-CPA Security of the Tester

To prove that this construction satisfies IND-CPA, we define a sequence of security games.

Game$_0$ This is the regular IND-CPA security game for our construction:

1. The challenger runs $\mathsf{Setup}(1^n) \to \text{PP} = (m, q, \ell, \sigma, r, \alpha, z)$.
2. The challenger chooses a uniformly random secret key sk in $\{0,1\}^z$ and runs TrapGen(sk) to produce a matrix $B \in \mathbb{Z}_q^{n \times m}$ and a corresponding trapdoor basis T_B. It then chooses independent and uniformly random vectors $c_1, \ldots, c_\ell \in \mathbb{Z}_q^n$ and noise vectors $\gamma_1, \ldots, \gamma_\ell$ from χ^m. It sets
$$pk = \{c_1, \ldots, c_\ell, y_1 := c_1 B + \gamma_1, \ldots, y_\ell := c_\ell B + \gamma_\ell\}.$$
The challenger gives the parameters PP and key pk to the attacker.
3. $\mathcal{A}$ The attacker submits two messages m_0, m_1 to the challenger.
4. The challenger flips a coin $b \in \{0,1\}$. It runs TrapGen(m_b) to produce a matrix $Z \in \mathbb{Z}_q^{n \times m}$ and a corresponding trapdoor basis T_Z. It chooses random signs $r_1, \ldots, r_\ell \in \{-1, 1\}$ and computes $s := \sum_{i=1}^{\ell} r_i c_i$. It then uses T_Z to sample a short (column) vector v such that $Zv = s^t$, by calling the algorithm SampleD. It computes
$$C = \sum_{i=1}^{\ell} r_i y_i,$$
and sets the ciphertext as (C, v).
5. The attacker receives the challenge ciphertext. It then outputs a guess b' and wins if $b' = b$.

Game$_1$

2. The challenger chooses a uniformly random secret key sk in $\{0,1\}^z$ and runs TrapGen(sk) to produce a matrix $B \in \mathbb{Z}_q^{n \times m}$ and a corresponding trapdoor basis T_B. It then chooses independent and uniformly random vectors $c_1, \ldots, c_\ell \in \mathbb{Z}_q^n$ and uniformly random vectors $y_1, \ldots, y_\ell \in \mathbb{Z}_q^m$. It sets $pk = \{c_1, \ldots, c_\ell, y_1, \ldots, y_\ell\}$.

Game$_2$

4. The challenger flips a coin $b \in \{0,1\}$. It runs TrapGen(m_b) to produce a matrix $Z \in \mathbb{Z}_q^{n \times m}$ and a corresponding trapdoor basis T_Z. It chooses s randomly in $\mathbb{Z}_q^n$. It then uses T_Z to sample a short (column) vector v such that $Zv = s^t$, by calling the algorithm SampleD. It chooses C randomly from $\mathbb{Z}_q^m$ and sets the ciphertext as (C, v).

Game$_3$

4. The challenger samples the vector v from $\mathcal{D}_{\mathbb{Z}^m, \sigma}$. It chooses C randomly from $\mathbb{Z}_q^m$ and sets the ciphertext as (C, v).

At this point, the distribution of the ciphertext is independent of the message, and it is clear that no PPT adversary can obtain a non-zero advantage.

Lemma 7.1 *Under the LWE assumption for the noise distribution χ, no PPT attacker can obtain a non-negligible difference in advantage between Game$_0$ and Game$_1$.*

Proof. We can collect the column vectors $c_1^t, \ldots, c_\ell^t$ into a $n \times \ell$ matrix we call D. We can collect the row vectors $y_1, \ldots, y_m$ into a $\ell \times m$ matrix we call Y and the row vectors $\gamma_1, \ldots, \gamma_\ell$ into a $\ell \times m$ matrix we call Γ. We can then write the public key as $D, D^t B + \Gamma$. Since B is never published, each column of B is a fresh, uniform vector in $\mathbb{Z}_q^n$, and therefore each column of $D^t B + \Gamma$ is distributed as an LWE sample with D playing the role of the $n \times m$ matrix A and the column of B playing the role of the random vector s. By a hybrid argument over the columns, we can thus rely on LWE to change each y_i to be uniformly distributed in $\mathbb{Z}_q^m$. □

Lemma 7.2 *No PPT attacker can obtain a non-negligible difference in advantage between Game$_1$ and Game$_2$.*

Proof. For this, we will argue that the distributions of s, C in Game$_1$ and Game$_2$ are statistically close. This is a direct application of lemma 2.2 with j set to be $n + m$. To see this, we consider the random signs $r_1, \ldots, r_\ell \in \{-1, 1\}$ as a column vector R of length ℓ. We then consider the (vertical) concatenation of s^t and C^t into a $n + m$ length column vector. In Game$_1$, this is produced as MR, where M is a $(n + m) \times \ell$ matrix formed by vertically concatenating D and Y^t as defined in the proof of the previous lemma. Since the matrices D, Y are now uniformly chosen, replacing MR by a uniformly random $(n + m) \times 1$ matrix (as in Game$_2$) is a statistically close distribution by Lemma 2.2. □

Lemma 7.3 *No PPT attacker can obtain a non-negligible difference in advantage between Game$_2$ and Game$_3$.*

Proof. We will argue that the distributions of v in Game$_2$ and Game$_3$ are statistically close. We first observe that in Game$_2$, v is chosen so that $Zv = s^t$ for a uniformly random s that is now independent of the rest of the ciphertext. The distribution of v here produced by SampleD is statistically close to $\mathcal{D}_{\Lambda_q^s(Z),\sigma}$ (relying on TrapGen succeeding in generating a sufficiently good trapdoor with probability 1). Now by Lemma 2.1, if we consider the distribution $\mathcal{D}_{\mathbb{Z}^m,\sigma}$, the probability mass on the preimages of s^t under the mapping $Zv = s^t$ is (up to a negligible statistical distance) the same for each s. Thus, the distribution of v in both Game$_2$ and in Game$_3$ is statistically close to $\mathcal{D}_{\mathbb{Z}^m,\sigma}$.

□

Acknowledgments

The authors thank Christina Garman and Matthew Green for helpful discussions.

We also thank Chris Peikert for making the important observation that Lemma 7.3 requires TrapGen to succeed with probability 1 in producing a sufficiently good trapdoor, and for noting that this can be obtained from [27].

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

[4] Miklós Ajtai. Generating hard instances of the short basis problem. In *Automata, Languages and Programming, 26th International Colloquium, ICALP'99, Prague, Czech Republic, July 11-15, 1999, Proceedings*, pages 1–9, 1999.

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

[6] Joël Alwen and Chris Peikert. Generating shorter bases for hard random lattices. In *26th International Symposium on Theoretical Aspects of Computer Science, STACS 2009, February 26-28, 2009, Freiburg, Germany, Proceedings*, pages 75–86, 2009.

[7] 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.

[8] 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.

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

[10] Karyn Benson, Hovav Shacham, and Brent Waters. The k-BDH Assumption Family: Bilinear Map Cryptography from Progressively Weaker Assumptions. In *Topics in Cryptology - CT-RSA*, pages 310–325, 2013.

[11] 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.

[12] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *CRYPTO '04*, volume 3152 of LNCS, pages 45–55, 2004.

[13] 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.

[14] 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.

[15] 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.

[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] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM J. Computing*, 30(2):391–437, 2000.

[19] 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.

[20] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 197–206, 2008.

[21] S. Goldwasser and S. Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[22] Iftach Haitner and Thomas Holenstein. On the (im)possibility of key dependent encryption. In *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*, pages 202–219, 2009.

[23] Dennis Hofheinz and Eike Kiltz. Secure hybrid encryption from weakened key encapsulation. In *Advances in Cryptology - CRYPTO*, pages 553–571, 2007.

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

[25] 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.

[26] 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.

[27] 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.

[28] Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. *SIAM J. Comput.*, 41(4):772–814, 2012.

[29] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC '90*, pages 427–437, 1990.

[30] 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.

[31] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *CRYPTO '91*, volume 576 of LNCS, pages 433–444, 1991.

[32] 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.

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

[34] Hovav Shacham. A cramer-shoup encryption scheme from the linear assumption and from progressively weaker linear variants. *IACR Cryptology ePrint Archive*, 2007:74, 2007.

A A 2-Cycle Tester from Decision Linear

We now present a 2-cycle tester from the Decision Linear assumption in symmetric pairing groups. This construction can be viewed as an extension of the SXDH one from Section 5 to symmetric groups where new variables and equations are introduced to prevent the use of pairings to disrupt IND-CPA security. However, it does not seem to generalize to a system that is secure using the decision k-linear assumption for $k > 2$ or help move toward a Learning with Errors Assumption.

At the same time when compared to our more general construction for the $k = 2$ (decision linear assumption) case, it achieves smaller public keys. Public keys are here are two group elements as opposed to four. We also believe that the techniques applied here might find use elsewhere in converting from constructions that rely on SXDH to those that do not.

Like in Section 5 we slightly abuse notation and let our message space $\mathbb{Z}_p^*$ be defined by our Setup algorithm. Our construction is below.

$\mathsf{Setup}(1^\lambda) \to \mathrm{PP}$. Recall that we assume a setting where all parties implicitly use shared public parameters. Run $\mathcal{G}(1^\lambda)$ to generate a Type-I pairing description $\mathrm{PP} = (p, g, \mathbb{G}, \mathbb{G}_T, e)$, such that p is prime in $\Theta(2^\lambda)$, $\mathbb{G}$ is a group of order p where g generates $\mathbb{G}$, and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is an admissible pairing. Choose random $a, b \in \mathbb{Z}_p^*$. Publish

$$\mathrm{CRS} = g, g^a, g^b, g^{ab}$$

The message space is $\mathbb{Z}_p^*$.

$\mathsf{KeyGen}(\mathrm{PP}) \to (pk, sk)$. Flip a coin $\beta \in \{0, 1\}$ and choose random $s \in \mathbb{Z}_p^*$. In addition to implicitly including PP, set the public key as:

$$pk = \begin{cases} (0, K_1 = g^{as}, K_2 = g^s) \in \{0,1\} \times \mathbb{G}^2 & \text{if } \beta = 0; \\ (1, K_1 = g^{bs}, K_2 = g^s) \in \{0,1\} \times \mathbb{G}^2 & \text{if } \beta = 1. \end{cases}$$

The secret key is $sk = s$, making the secret key space $\mathbb{Z}_p^*$ (which is the same as the message space).

$\mathsf{Enc}(pk, m \in \mathbb{Z}_p^*) \to C$. To encrypt, parse $pk = (\beta, K_1, K_2)$ and choose random $t, r \in \mathbb{Z}_p^*$. Generate the ciphertext $C = (C_1, C_2, C_3, C_4) \in \mathbb{G}^4$ as:

$$C = \begin{cases} (C_1 = K_1^{tm} = g^{astm}, C_2 = (g^a)^{t+r}, C_3 = (g^{ab})^r, C_4 = K_2^{tm} = g^{stm}) & \text{if } \beta = 0; \\ (C_1 = K_1^{tm} = g^{bstm}, C_2 = (g^b)^{t+r}, C_3 = (g^{ab})^r, C_4 = K_2^{tm} = g^{stm}) & \text{if } \beta = 1. \end{cases}$$

$\mathsf{Test}(\mathbf{pk}, \mathbf{y}) \to \{0, 1\}$. Since we are testing for 2-cycles, parse $\mathbf{pk} = (pk = (\beta, K_1, K_2), pk' = (\beta', K_1', K_2'))$ and $\mathbf{y} = (C = (C_1, C_2, C_3, C_4), C' = (C_1', C_2', C_3', C_4'))$. If $\beta = \beta'$, output a random bit. Otherwise, assume without loss of generality that $\beta = 0$ and $\beta' = 1$. Then the test algorithm checks if

$$\frac{e(C_1, C_2')}{e(C_4, C_3')} \stackrel{?}{=} \frac{e(C_1', C_2)}{e(C_4', C_3)}$$

If this check holds, then output 1 (guess cycle); otherwise output 0 (guess zeros).

Relation to SXDH example. To relate this construction to the one in Section 5, note that one can reverse derive the asymmetric version from this symmetric one by letting $r = 0$, letting $a = b = 1$ and then placing variables into the group $\mathbb{G}_1$ or $\mathbb{G}_2$ depending on their input position in the pairings.

Analysis of Test Algorithm In the IND-CIRC-CPA^2 game, two public keys $pk = (\beta, K_1, K_2), pk' = (\beta', K_1', K_2')$ are freshly generated by the KeyGen algorithm according to common parameters output by the Setup algorithm. Since β and β' are independently chosen bits, the probability that $\beta = \beta'$ is 1/2. In this case, the Test algorithm will output a random bit (i.e., a random guess) and thus will be successful exactly 1/2 the time.

Otherwise, we assume without loss of generality that $\beta = 0$ and $\beta' = 1$ (if not, swap the two key/ciphertext pairs). In this game, the challenge ciphertexts (C, C') are formed using the Enc algorithm. By plugging these values into the Test equation above, we have:

$$\begin{aligned}
\frac{e(g^{astm}, g^{b(t'+r')})}{e(g^{stm}, g^{abr'})} &= \frac{e(g^{bs't'm'}, g^{a(t+r)})}{e(g^{s't'm'}, g^{abr})} \\
\frac{e(g,g)^{abstm(t'+r')}}{e(g,g)^{abstmr'}} &= \frac{e(g,g)^{abs't'm'(t+r)}}{e(g,g)^{abs't'm'r}} \\
e(g,g)^{abstm(t'+r')-abstmr'} &= e(g,g)^{abs't'm'(t+r)-abs't'm'r} \\
e(g,g)^{abstmt'} &= e(g,g)^{abs't'm't} \\
e(g,g)^{sm} &= e(g,g)^{s'm'}
\end{aligned}$$

From the last equation, it is clear that if there is a cycle ($m = s'$ and $m' = s$), then both sides equal $e(g,g)^{ss'}$ and the equation holds. If we have an encryption of zero (recall this is encoded as some element $z \in \mathbb{Z}_p^*$, so $z \neq 0$), then the left-hand-side is $e(g,g)^{sz}$ and the right-hand-side is $e(g,g)^{s'z}$, which will only be equal if $s = s'$. Since these secret keys were randomly chosen from $\mathbb{Z}_p^*$, the chance they are the same is $1/(p-1)$, which is negligible in λ. Thus, the success of the Test algorithm is (1/2)(1/2) + (1/2)((1/2) 1 + (1/2) (1-1/(p-1)) = 3/4 - negl(λ), which easily wins the IND-CIRC-CPA^2 game.

IND-CPA Security of the Tester

Theorem A.1 *The above encryption scheme* $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \cdot)$ *(where the decryption algorithm is ignored) is* IND-CPA*-secure under the Decision Linear Assumption in* $\mathbb{G}$.

Proof. Suppose that there exists an algorithm $\mathcal{A}$ that has a non-negligible advantage ϵ in the IND-CPA security game. We can then construct an algorithm $\mathcal{B}$ that solves the decision linear problem with advantage $\frac{1}{2} + \frac{\epsilon}{2}$. $\mathcal{B}$ works as follows:

1. Select a random bit $\beta \in \{0,1\}$. We proceed assuming $\beta = 0$; the case of $\beta = 1$ is analogous.
2. Obtain a decision linear instance $(g, g_1, g_2, g_1^{r_1}, g_2^{r_2}, T_d)$.
3. Choose a random $a \in \mathbb{Z}_p^*$ and publish
$$\text{CRS} = (g, g^a, g^b = g_2, g^{ab} = g_2^a).$$
4. Set the public key pk as
$$pk = (0, K_1 = g^{as} = g_1^a, K_2 = g^s = g_1)$$
and output it to $\mathcal{A}$. Note that $\mathcal{B}$ does not know the secret key.
5. Run $\mathcal{A}(pk)$ to produce a tuple (m_0, m_1).
6. Select a random bit $\gamma \in \{0,1\}$.
7. Set the challenge ciphertext as
$$C_1 = K_1^{tm_\gamma} = (g_1^{r_1})^{am_\gamma}, C_2 = T_d^a, C_3 = (g^{ab})^r = (g_2^{r_2})^a, C_4 = K_2^{tm_\gamma} = (g_1^{r_1})^{m_\gamma}.$$
(implicitly setting $t = r_1$ and $r = r_2$) and output it to $\mathcal{A}$.
8. Run $\mathcal{A}(C)$ to get a bit γ'. If $\gamma = \gamma'$, then output 0 (guess a DLIN instance). Otherwise, output 1 (guess T_d was random).

If $T_d = g^{r_1+r_2}$, then $C_2 = T_d^a = (g^{r_1+r_2})^a = (g^a)^{t+r}$ and the ciphertext is well-formed according to the original encryption algorithm. $\mathcal{B}$ wins with probability $\frac{1}{2} + \epsilon$ in this case. Otherwise, C_2 is a random group element, which means that $\mathcal{A}$ guesses correctly with probability $\frac{1}{2}$. In this case, $\mathcal{B}$ wins with probability $\frac{1}{2}$ as well. $\mathcal{B}$ then has an overall advantage of $\frac{1}{2}\left(\frac{1}{2} + \epsilon\right) + \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2} + \frac{\epsilon}{2}$, which is non-negligible.

□