Toward Fine-Grained Blackbox Separations Between Semantic and Circular-Security Notions

Mohammad Hajiabadi[1,2(✉)] and Bruce M. Kapron[1,2]

[1] Department of Computer Science, University College London, London, UK
m.hajiabadi@ucl.ac.uk
[2] Department of Computer Science, University of Victoria, Victoria, Canada
bmkapron@uvic.ca

Abstract. We address the problems of whether t-circular-secure encryption can be based on $(t-1)$-circular-secure encryption or on semantic (CPA) security, if $t = 1$. While for $t = 1$ a folklore construction, based on CPA-secure encryption, can be used to build a 1-circular-secure encryption with the same secret-key and message space, no such constructions are known for the bit-encryption case, which is of particular importance in fully-homomorphic encryption. Also, all constructions of t-circular encryption (bitwise or otherwise) are based on specific assumptions.

We make progress toward these problems by ruling out all fully-blackbox constructions of

- 1-*seed-circular-secure* bit encryption from CPA-secure encryption;
- t-seed-circular-secure encryption from $(t-1)$-seed-circular secure encryption, for any $t > 1$.

Informally, seed-circular security is a variant of the circular security notion in which the seed of the key-generation algorithm, instead of the secret key, is encrypted. We also show how to extend our first result to rule out a large and non-trivial class of constructions of 1-circular-secure bit encryption, which we dub *key-isolating constructions*. Our separations follow the model of Gertner, Malkin and Reingold (FOCS'01), which is a weaker separation model than that of Impagliazzo and Rudich.

1 Introduction

A public-key encryption scheme is 1-circular secure if it is CPA secure in the presence of an encryption of the secret key under its corresponding public key. A more general notion is that of t-circular security under which CPA security under t public keys $pk_0, \ldots, pk_{t-1}$ should be maintained even when each pk_i is used to encrypt the secret key of $pk_{(i+1 \bmod t)}$. These notions are a special case of the general notion of key-dependent-message (KDM) security, under which more general functions of the secret key(s) may be encrypted.

Work supported in part by the NSERC Discovery Grant "Foundational Studies in Privacy and Security". Part of this work completed while the first author was at University College London and received funding from the European Research Council under the ERC Grant Agreement no. 307937.

J.-S. Coron and J.B. Nielsen (Eds.): EUROCRYPT 2017, Part II, LNCS 10211, pp. 561–591, 2017.
DOI: 10.1007/978-3-319-56614-6_19

A primary foundational application of the notion of circular security (for any t) is in the context of fully homomorphic encryption. Currently, with the exception of [11], all constructions of *pure fully homomorphic* encryption go through a *bootstrapping* procedure, requiring a circular-security assumption on a *bootstrappable* scheme built along the way.

When discussing circular security for an encryption scheme with secret-key space $\{0,1\}^\tau$ and plaintext space $\{0,1\}^\eta$, an important feature is the relation between τ and η: we call a scheme *full-length* if $\tau = \eta$. It is straightforward to build a full-length 1-circular-secure scheme from any CPA-secure scheme.[1] Informally, this folklore construction is based on the idea that the underlying plaintext m and public key pk can "communicate" to see if m is pk's secret key. Attempts in extending this idea to the t-circular security setting (for $t > 1$) have so far met with less success and in fact to date all constructions of t-circular secure schemes (full-length or otherwise) are based on assumptions with certain algebraic properties or obfuscation assumptions [3,7,9,30,31,43]. One of the goals of our work is to explain this state of difficulty.

Unfortunately, the full-length assumption is not the end of the story since in many applications of circular security, the secret key is indeed encrypted bit-by-bit or block-by-block, where the size of each block is considerably smaller than the secret-key size (e.g., [16,42]). In such cases the above folklore construction (for $t = 1$) no longer applies: the main difficulty is that since the secret key is no longer encrypted as a whole, but as short blocks, we cannot perform the simple check described above. Of particular importance in such settings is the notion of circular security for *single-bit* encryption schemes (which we call *bit-circular security*), which, beyond FHE applications, is fundamental for the following reason: as shown by Applebaum [2], *projection security*, a notion slightly extending bit-circular security by also allowing for encryptions of negated secret-key bits, is sufficient to obtain KDM security w.r.t. any (*a priori* fixed) function family. Thus, understanding basic forms of KDM security in the bitwise setting is essential for the general understanding of KDM security.

Toward understanding the notion of circular security, several papers based on various specific assumptions have given schemes that are CPA secure, but not t-circular secure (for various values of t), [1,6,12,27]. We remark that although these works provide evidence that t-circular security of any scheme cannot be reduced to the CPA security of the same scheme, they do not shed light on the impossibility of positive constructions.

Finally, we mention that despite the foundational importance of the notion of bit-circular security, our understanding of what it takes to obtain this notion (without relying on specific assumptions) is still lacking, and there is little previous addressing the problem. Haitner and Holenstein [21] rule out fully-blackbox constructions of KDM-secure encryption w.r.t. quite large function families from

[1] Assume, w.l.o.g, the CPA-secure scheme (G, E, D) has plaintext space $\{0,1\}^n$ and that G uses an n-bit seed, which is also the outputted secret key. Briefly, the idea is to modify E so that $E(pk, m)$ will first check whether $G(m)$ produces pk as the public key, in which case it returns an encryption of an innocuous message.

trapdoor permutations. Rothblum [39] shows no fully-blackbox reduction can prove that CPA security of a bit-encryption scheme implies circular security of the same scheme. We stress that the result of [39] only considers reductions to and from the same scheme, as opposed to the results of this paper which are concerned with constructions.

Before moving on, we remind the reader of the simple fact that bit t-circular security implies full-length t-circular security. Briefly, the state of knowledge regarding circular security can be summarized as follows:

- Full-length t-circular security based on CPA security: we have a simple construction for $t = 1$, but no known constructions for $t > 1$.
- For bit t-circular security: all constructions (for $t = 1$ or beyond) are based on specific assumptions [7,9] and there is a preliminary separation for $t = 1$ from CPA security [39].

In this work we ask the following two questions

(1) Can bit 1-circular security be based on CPA security?
(2) Can full-length t-circular security (for $t > 1$) be based on CPA security?

1.1 Our Contributions and Discussion

In this paper we make progress toward answering both questions above in the negative, by considering the stronger notion of *seed-circular* security. In its simplest form, an encryption scheme is 1-seed-circular secure if it is CPA secure in the presence of an encrypted version of the seed (of the key-generation algorithm) under its corresponding public key. Similarly, we may define bit/full-length) t-seed circular security. Note that the assumption of t-seed-circular security is indeed at least as strong as that of t-circular security since any scheme meeting the former can slightly be changed to meet the latter by altering the key-generation algorithm to return the underlying seed as its secret-key output. We first describe our main results and then discuss them in detail.

1. We prove there exists no fully blackbox construction (in the sense of [36]) of 1-*seed-circular-secure* bit encryption from CPA-secure encryption (Theorem 6.) We also show that this separation holds so long as the constructed scheme has plaintext space $\{0,1\}^{c \log n}$ for any constant c (Sect. 5.6).
2. We prove that full-length $(t+1)$-seed-circular security cannot be based in a fully-blackbox way on *bit* t-seed circular security, for any $t \geq 1$ (Theorem 9).

Our first result already rules out certain types of constructions for 1-circular-secure encryption, namely those in which seeds and secret keys are the same. We show how to adapt this result to the setting of circular security, to rule out a large and non-trivial class of constructions of circular-secure encryption that we call *key-isolating constructions*. Due to technicalities involved we refer the reader to Sect. 7 for this notion. (A similar adaptation may be given for the second result, but we do not pursue it in this paper.)

For our second result, choosing the target notion to be full-length $(t+1)$-seed circular security (as opposed to bit $(t+1)$-seed circular security) and the base notion to be bit t-seed circular security (as opposed to full-length t-seed-circular security) only makes our result stronger.

Discussion of results and notions. We first start by discussing the significance of the second result. We note that the folklore CPA-security-based construction alluded to earlier indeed results in a full-length 1-seed-circular secure scheme, since the constructed scheme has the same seed and secret-key space. This shows that the notion of seed-circular security (at least for the full-length case) is not so far fetched, reinforcing the significance of the separation result and providing partial justification for the lack of success in basing full-length t-circular security, for $t > 1$, on CPA security. In fact, it suggests that a less ambitious goal than that of Question (2), namely of basing t-circular security on $(t-1)$-circular security, may still be too much to hope for.

As for the first result, we mention the following fact regarding the notion of bit 1-seed-circular security. Since one of the main applications of this notion is in the context of FHE, it is worth mentioning that if $\mathcal{E}$ is fully homomorphic (or homomorphic enough to evaluate G), then if $\mathcal{E}$ is 1-seed-circular secure it is also 1-circular secure, since one can use the homomorphic properties of $\mathcal{E}$ to evaluate G homomorphically, thereby producing an encrypted secret key from an encrypted seed. (This simple proof is, however, non-blackbox.)

From a practical point of view, the notion of seed circular security for specific schemes is not very natural since such schemes typically come with *public parameters* (e.g., a group), and it is not very meaningful to talk about, say, encrypting the bits used to generate those parameters. Nevertheless, if public-parameter generation is thought of as a separate process, many specific schemes have the property that their secret keys are just the same as their seeds (e.g., ElGamal). For example, both circular-secure schemes of [7,9] have the property that w.r.t. fixed public parameters (which are a group plus l group elements), their secret keys are just random l-bit-strings, being the same as their seeds. Thus, as a step toward proving full blackbox impossibility for circular-secure encryption, it may be worthwhile to formulate a notion of encryption with public parameters, and investigate whether our results extend to this case. We have not, however, carried this out at this moment.

We conclude the discussion with the following observation. Our first result leaves us with an unexplained gap, namely to what extent the plaintext size of the constructed scheme could be made bigger before obtaining a positive (seed-)circular security result? For example, what happens if the construction is allowed to have plaintexts of $\omega(\log n)$ bits long? We believe that filling this gap will further improve our understanding of the notion of 1-(seed-)circular security.

Our separation model. All our separations follow the model of [19]. We discuss the model for the first result. For any candidate 1-seed-circular-secure construction $\mathcal{E} = (G, E, D)$ we show the existence of two oracles $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d})$ and $\mathbf{T}$ such that (a) there exists a PPT oracle adversary $\mathcal{A}^{\mathbf{O},\mathbf{T}}$ that breaks the (supposed)

seed-circular security of $\mathcal{E}^{\mathbf{O}}$ and (b) no PPT oracle adversary $\mathcal{B}^{\mathbf{O},\mathbf{T}}$ can break the CPA security of $\mathbf{O}$. This immediately implies that there exists no fully-blackbox reduction. As common in separation models we show the existence of $\mathbf{O}$ and $\mathbf{T}$ non-constructively by proving results w.r.t. randomly chosen $\mathbf{O}$ and $\mathbf{T}$. We give an overview of our techniques and separation model in Sect. 4.

Most separation results in the literature indeed rule out the existence of *relativizing reductions*, e.g., [8,17,25,40,41], which constitute a broader class of constructions than fully-blackbox ones. We stress that our results do not rule out relativizing reductions. Nonetheless, we are not aware of any "natural" cryptographic construction that is relativizing but not fully-blackbox. Finally, we mention that there exists separation results in the literature that also only rule out fully-blackbox reductions, e.g., [21,24,28,29].

Blackbox versus non-blackbox techniques. We note that there are non-blackbox reductions in cryptography, for which a blackbox-counterpart may or may not (both provably and ostensibly) exist. (Here by non-blackbox we are referring to the construction, not to the security proof.) We mention [14,26] as two examples of blackbox constructions that replaced their earlier non-blackbox counterparts [20,35]. Classical examples of non-blackbox constructions with no known blackbox counterparts are [15,34], giving non-blackbox constructions of CCA1- and CCA2-secure encryption from *enhanced trapdoor permutations*. The state of our knowledge regarding the blackbox status of CCA-secure encryption versus other "classical" public-key primitives is arguably limited, and the only known works are the work of Gertner *et al.* [18], ruling out *sheilding* blackbox constructions of CCA1-secure encryption from CPA-secure encryption, and that of Myers and shelat [33] proving equivalence of one-bit and many-bit CCA2 secure encryption. Finally, we mention that the work of Mahmoody and Pass [29] shows the existence of a non-blackbox construction (that of non-interactive commitment schemes from so called *hitting one-way functions*) for which provably no blackbox counterpart exists.

Other related work. The question of what "general" assumptions may be used to obtain KDM security is addressed in [23], where it is shown that projection-secure public-key encryption (PKE) can be built from any CPA-secure PKE with some structural properties. The power of circular-secure encryption is addressed in [22], where it is shown that in combination with a so-called reproducibility property, bit circular security implies the existence of powerful primitives including correlation-secure trapdoor functions [38], CCA2-secure encryption and deterministic encryption. The body of work on blackbox separations is extensive, some of which were mentioned earlier. We also mention the progress that has been made in understanding the limitations of some of the common non-blackbox techniques, e.g., [4,10].

Open Problems. The main open problem is to extend our impossibility results to the circular-security setting. We explain in Sect. 8 why we were not able to do this. Another interesting problem is to see to what extent our techniques extend to obtain separations based on other classical public-key primitives.

Note on proofs. Due to space constraints, proofs for some results have been omitted. In all cases, proofs for these results appear in the full version.

2 Preliminaries

If $R(x_1, \ldots, x_i; r)$ is a randomized algorithm using randomness r, by $R(a_1, \ldots, a_i)$ we mean the random variable obtained by sampling r uniformly at random and returning $R(a_1, \ldots, a_i; r)$. If $\mathcal{D}$ is a distribution $x \in \mathcal{D}$ means $x \in \mathit{support}(\mathcal{D})$.

The notion of a public-key encryption scheme (PKE) (G, E, D) is standard. The only convention we make is that the order of keys produced by G is as a secret/public key pair (as opposed to a public/secret key pair). We refer to the randomness space of G as the *seed* space of the scheme. We assume the decryption algorithm is deterministic, and always decrypts correctly, and refer to this as the *correctness* condition. (Our separation results will hold even if the constructed scheme is allowed to make a small decryption error. However, for the sake of simplicity we assume the stated condition.) All schemes in this paper are many-bit or single-bit encryption schemes. If E's plaintext space is $\{0,1\}^\eta$ by $E(PK, M)$ for $M \in \{0,1\}^*$ we mean that M is encrypted in blocks of size η, augmenting M with enough zero bits to make $|M|$ a multiple of η, if necessary. In particular, when $\eta = 1$, this will denote the bit-by-bit encryption of M.

We shall use lowercase letters $(\mathbf{g}, \mathbf{e}, \mathbf{d})$ to denote base (i.e., blackbox) schemes and uppercase letters (G, E, D) to denote constructions.

Oracle convention. Whenever we talk about an oracle adversary/algorithm $\mathcal{A}$ we adopt the following conventions: we say $\mathcal{A}$ is *efficient* (or PPT) if $\mathcal{A}$ can be implemented as a PPT oracle algorithm; we say $\mathcal{A}$ is *query-efficient* if $\mathcal{A}$ always makes at most a poly-number of oracle queries (but unlimited otherwise, and may run exponential local computations). Whenever we put no restriction on an adversary it means that it is not restricted in any way.

We define when an adversary breaks the (seed-)circular security of a bit-encryption scheme. The definition naturally extends to the many-bit case.

Definition 1. *Let $\mathcal{E} = (G, E, D)$ be a bit PKE with seed space $\{0,1\}^n$. Let*

$$\mathbf{InpSeed} = (PK_1, \ldots, PK_t, E_{PK_1}(S_2), \ldots, E_{PK_{t-1}}(S_t), E_{PK_t}(S_1))$$
$$\mathbf{InpSec} = (PK_1, \ldots, PK_t, E_{PK_1}(SK_2), \ldots, E_{PK_{t-1}}(S_t), E_{PK_t}(SK_1))$$
$$b \leftarrow \{0,1\}, C \leftarrow E_{PK_1}(b),$$

where $S_i \leftarrow \{0,1\}^n$ and $(SK_i, PK_i) = G(S_i)$, for $1 \leq i \leq t$. Then we say

- *$\mathcal{A}$ breaks the t-seed-circular security of $\mathcal{E}$ if $\Pr[\mathcal{A}(\mathbf{InpSeed}, C) = b]$ is non-negligibly greater than $1/2$.*
- *$\mathcal{A}$ breaks the t-circular security of $\mathcal{E}$ if $\Pr[\mathcal{A}(\mathbf{InpSec}, C) = b]$ is non-negligibly greater than $1/2$.*

We now define the assumptions underlying our results in this paper.

Terminology 1. The assumption of "bit t-seed-circular security" refers to the existence of a t-seed-circular secure single-bit PKE. Also, "full-length t-seed-circular security" refers to the existence of a t-seed-circular secure PKE with the same seed and plaintext space. We have the following simple implications: (a) CPA security $\Rightarrow$ full-length 1-seed circular security and (b) bit t-seed-circular security $\Rightarrow$ full-length t-seed-circular security.

We define a notion of blackbox reductions between encryption primitives. See [5,36] for more general notions of blackbox reductions.

Definition 2. *A* fully-blackbox reduction of *P-secure (e.g., circular-secure) encryption to Q-secure (e.g., CPA-secure) encryption consists of two PPT oracle algorithms $(\mathcal{E}, Red)$, satisfying the following: for any PKE $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d})$,*

1. *$\mathcal{E}^{\mathbf{O}} = (G^{\mathbf{O}}, E^{\mathbf{O}}, D^{\mathbf{O}})$ forms a PKE, and*
2. *for any adversary $\mathcal{A}$ breaking the P-security of $(G^{\mathbf{O}}, E^{\mathbf{O}}, D^{\mathbf{O}})$, the oracle algorithm $Red^{\mathcal{A},\mathbf{O}}$ breaks the Q-security of $\mathbf{O}$.*

3 PKE Oracle Distribution

Convention. Whenever we say a function $f\colon D \to R$ with property P (e.g., injectivity) is a randomly chosen function we mean f is chosen uniformly at random from the space of all functions from D to R having property P.

We describe a distribution under which a PKE oracle (with some auxiliary oracles) is sampled. These oracles will be used to model "ideal" base primitives in our separations. We largely follow the notational style of [18]. As notation, if f is a function whose output is a tuple, say a pair, we write $f(x) = (*, y)$ to indicate that $f(x) = (y', y)$, for some y'.

Definition 3. *We define an oracle distribution Ψ which produces a PKE oracle with certain length parameters, plus two auxiliary oracles. Formally, Ψ produces an ensemble of oracles $\mathcal{O}_n = (\mathbf{O}_n, \mathbf{u}_n, \mathbf{w}_n)_{n\in\mathbb{N}}$, where for every $n \in \mathbb{N}$, $\mathbf{O}_n = (\mathbf{g}_n, \mathbf{e}_n, \mathbf{d}_n)$ and $(\mathbf{u}_n, \mathbf{w}_n)$ are chosen as follows.*

- $\mathbf{g}_n\colon \{0,1\}^n \to \{0,1\}^{5n}$ *is a random one-to-one function, mapping a secret key to a public key.*
- $\mathbf{e}_n\colon \{0,1\}^{5n} \times \{0,1\} \times \{0,1\}^n \to \{0,1\}^{7n}$ *is a function, where for every $pk \in \{0,1\}^{5n}$, $\mathbf{e}_n(pk, \cdot, \cdot)$ is a random one-to-one function.*
- $\mathbf{d}_n\colon \{0,1\}^n \times \{0,1\}^{7n} \to \{0,1\} \cup \{\perp\}$ *is defined by letting $\mathbf{d}_n(sk, c) = b$ if and only if $\mathbf{e}_n(\mathbf{g}_n(sk), b, r) = c$, for some $r \in \{0,1\}^n$; otherwise, $\mathbf{d}_n(sk, c) = \perp$.*
- $\mathbf{u}_n\colon \{0,1\}^{5n} \times \{0,1\}^{7n} \to (\{0,1\} \times \{0,1\}^n) \cup \{\perp\}$ *is defined as $\mathbf{u}_n(pk, c) = (b, r)$ if $\mathbf{e}_n(pk, b, r) = c$, and $\mathbf{u}_n(pk, c) = \perp$ if for no (b, r) does it hold that $\mathbf{e}_n(pk, b, r) = c$. That is, $\mathbf{u}_n(pk, c)$ decrypts c relative to pk, and if successful, also returns the unique randomness used to produce c. (The oracle $\mathbf{u}$ is not typically allowed to be freely used. See Definition 4.)*
- $\mathbf{w}_n\colon \{0,1\}^{5n} \to \{\perp, \top\}$ *is defined as $\mathbf{w}_n(pk) = \top$ if for some sk $\mathbf{g}_n(sk) = pk$, and $\mathbf{w}_n(pk) = \perp$, otherwise. That is, $\mathbf{w}_n(pk)$ checks whether pk is a valid public key.*

Definition 4. *In all settings where access to* $\mathbf{u}$ *is granted this access is limited and is determined based on the underlying challenge inputs. Specifically, we call* $\mathcal{A}^{\mathbf{g},\mathbf{e},\mathbf{d},\mathbf{u},\mathbf{w}}$ *CCA-valid if* $\mathcal{A}^{\mathbf{g},\mathbf{e},\mathbf{d},\mathbf{u},\mathbf{w}}$ *on input* (pk, c) *never calls* $\langle \mathbf{u}, (pk, c)\rangle$*. This definition naturally generalizes to the case in which* $\mathcal{A}$*'s input consists of several challenge public keys with several challenge ciphertexts for each public key, e.g., the* t*-seed circular security setting.*

Omitting the security parameter. We define $\mathbf{g}(sk) = \mathbf{g}_n(sk)$, for every n and $sk \in \{0,1\}^n$, and use a similar convention for other functions in Definition 3. Sometimes when we need to emphasize under what security parameter a query is made, we put in the sub-index n; in other places we typically omit the sub-index.

Ψ-valid oracles. We call a triple of functions $(\mathbf{g}, \mathbf{e}, \mathbf{d})$ *Ψ-valid* if $(\mathbf{g}, \mathbf{e}, \mathbf{d})$ is part of a possible output of Ψ, i.e., the domains and ranges of $\mathbf{g}$, $\mathbf{e}$ and $\mathbf{d}$ are as specified in Definition 3, and also all the corresponding injectivity conditions hold. Similarly, we may use the same convention to call, say, $\mathbf{g}$, Ψ-valid.

Notation. For oracles $O = (O_1, \ldots, O_m)$ and an oracle algorithm $\mathcal{A}^O$, we let $qry = \langle O_i, q\rangle$ denote an $\mathcal{A}$'s query q to oracle O_i; if $u = O_i(q)$ we use $(\langle O_i, q\rangle, u)$ to indicate that $\mathcal{A}$ calls O_i on q and receives u; we also define $O(qry) = u$. If Que is a set of such *query/response pairs* we use shorthands like $(\langle O_j, *\rangle, u) \in \mathsf{Que}$ to mean that for some q, $(\langle O_j, q\rangle, u) \in \mathsf{Que}$. Thus, $(\langle O_j, *\rangle, u) \notin \mathsf{Que}$ indicates that for no q, we have $(\langle O_j, q\rangle, u) \in \mathsf{Que}$.

Symbolic representation of oracle queries. Sometimes we need to talk about sets containing query/response pairs generated under some oracle, and later on check them against another oracle. For this reason, we may sometimes talk about *symbolic* query/response pairs. For example, the symbolic form of a concrete query/response pair $(\langle \mathbf{g}, sk\rangle, pk)$ is denoted $(\langle \mathfrak{g}, sk\rangle, pk)$.

4 General Overview of Techniques

We give an overview of our approaches for the two main results: separating bit 1-seed circular security (see Terminology 1) from CPA security and separating full-length $(t+1)$-seed-circular security from bit t-seed-circular security.

4.1 CPA Security $\not\Rightarrow$ Bit 1-seed-circular Security

Summary of approach. First, note a random $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d})$, chosen as $(\mathbf{O}, \mathbf{u}, \mathbf{w}) \leftarrow \Psi$ will be "ideally" secure w.r.t. all notions security discussed in this paper. One idea for proving separations is to add some weakening components $\mathbf{v}$ to $\mathbf{O}$ and show that relative to $(\mathbf{O}, \mathbf{v})$ the base primitive exists, but not the target primitive. We could not make this approach work. Instead, we follow the model of [19], by defining a weakening oracle $\mathbf{T}$, for every candidate construction (G, E, D), in such a way that $\mathbf{T}$ breaks the claimed security of $(G^{\mathbf{O}}, E^{\mathbf{O}}, D^{\mathbf{O}})$, for a random $\mathbf{O}$, but not the base security of $\mathbf{O}$. We emphasize that $\mathbf{T}$ depends on (G, E, D).

Let $\mathcal{E} = (G, E, D)$ be a candidate bit-encryption construction, $(\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{u}, \mathbf{w}) \leftarrow \Psi$ and $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d})$. Our goal is to give an oracle $\mathbf{T}$ s.t. (I) $\mathbf{T}$ is helpful in breaking the (alleged) seed-circular-security of $\mathcal{E}^{\mathbf{O}}$ and (II) $\mathbf{T}$ is not helpful in breaking the CPA security of $\mathbf{O}$. The most obvious idea is that on inputs of the form $(PK, C_1, \ldots, C_n)$, an alleged public key PK and a bit-by-bit encryption of PK's seed under $\mathcal{E}^{\mathbf{O}}(PK, \cdot)$, $\mathbf{T}$ will check whether PK is a *valid* public key under $G^{\mathbf{O}}$ and if so decrypt $C_1, \ldots, C_n$ under a secret key corresponding to PK to get some string S and return S if $G^{\mathbf{O}}(S)$ produces PK.

There are two problems with the above raw approach. First, even doing a simple check, namely whether PK is a valid public key, can potentially grant a CPA adversary against $\mathbf{O}$ much power, violating Condition (II) above. (It is not hard to think of contrived constructions $\mathcal{E}$ for which this is the case.) Second, even if we assume a CPA-adversary $\mathcal{A}^{\mathbf{O},\mathbf{T}}(pk, c)$—against $\mathbf{O}$—always calls $\mathbf{T}$ on valid PK's, we still have to make sure that $\mathcal{A}$ cannot come up with a clever query $\mathbf{T}(PK, C_1, \ldots, C_n)$ whose response leaks information about $\mathbf{g}^{-1}(pk)$ or about c's plaintext bit.

Our approach starts by resolving the first problem, using an idea from [19] (also used in some subsequent works [18,41]): the oracle $\mathbf{T}$ performs the decryption of $(C_1, \ldots, C_n)$ not relative to $\mathbf{O}$, but relative to some $\widetilde{\mathbf{O}} = (\widetilde{\mathbf{g}}, \widetilde{\mathbf{e}}, \widetilde{\mathbf{d}})$, under which PK is indeed a valid public key (i.e., $\mathbf{T}$ decrypts using $D^{\widetilde{\mathbf{O}}}(SK', C_1 \ldots C_n)$, where $(SK', PK) \in G^{\widetilde{\mathbf{O}}}$). Without further restrictions on $\widetilde{\mathbf{O}}$ the result of decryption is most likely a random noise, as $\widetilde{\mathbf{e}}$ and $\widetilde{\mathbf{d}}$ can behave arbitrarily. Thus, we need to ensure that w.h.p. over a random R, $E^{\mathbf{O}}(PK, b; R) = E^{\widetilde{\mathbf{O}}}(PK, b; R)$, for any bit b. This would ensure that $D^{\widetilde{\mathbf{O}}}(SK', C_1 \ldots C_n)$ w.h.p. will be the real output, if $(PK, C_1, \ldots, C_n)$ were "honestly" generated, showing that $\mathbf{T}$ is useful in breaking 1-seed-circular security of $(G^{\mathbf{g}}, E^{\mathbf{e}}, D^{\mathbf{d}})$.

Specifically, we construct $\widetilde{\mathbf{O}}$ by *super-imposing* a poly number of query/response pairs Q_{s}, which serve as a *certificate* of PK's validity, on $\mathbf{O}$. More precisely, we first sample (offline) a set of query/response pairs Q_{s} in such a way that $G^{\mathsf{Q}_{\mathsf{s}}} = (*, PK)$; then, we super-impose (Definition 5) Q_{s} on $\mathbf{O}$ to obtain $\widetilde{\mathbf{O}}$. (Sometimes Q_{s} needs to also agree with some previous information.)

To resolve the second problem the oracle $\mathbf{T}$ will refuse to decrypt queries deemed "dangerous": those that can be issued by a CPA adversary $\mathcal{A}$ against $\mathbf{O}$, and whose responses may leak information about $\mathcal{A}$'s challenge secrets. The main challenge is to formulate these dangerous queries in such a way that $\mathbf{T}$ is provably of no use to any CPA adversary against $\mathbf{O}$, while guaranteeing that $\mathbf{T}$ still decrypts w.h.p. a randomly encrypted random seed chosen relative to $E^{\mathbf{O}}$.

Concrete overview. We now give a concrete overview of the above approach for a simple class of constructions. We first start by defining the task of super-imposing a set of $\mathfrak{g}$-type query/response pairs on an oracle $(\mathbf{g}, \mathbf{e}, \mathbf{d})$.

Definition 5. *We define the following procedure we call KeyImpose.*

– ***Input:*** $(\mathbf{g}, \mathbf{e}, \mathbf{d})$ *and a set* $\mathsf{Q}_{\mathsf{s}} = \{(\langle \mathfrak{g}, sk_1 \rangle, pk_1), \ldots, (\langle \mathfrak{g}, sk_w \rangle, pk_w)\}$*, satisfying* $sk_i \neq sk_j$ *for all distinct* i *and* j*.*

– ***Output:*** $(\widetilde{\mathbf{g}}, \widetilde{\mathbf{d}})$, *where*

$$\widetilde{\mathbf{g}}(sk) = \begin{cases} \mathbf{g}(sk) & \text{if } sk \notin \{sk_1, \ldots, sk_w\} \\ pk_i & \text{if } sk = sk_i \text{ for some } 1 \leq i \leq w \end{cases} \quad (1)$$

$\widetilde{\mathbf{d}}(sk, c)$ *is defined as follows: if there exist* b *and* r *such that* $\mathbf{e}(\widetilde{\mathbf{g}}(sk), b, r) = c$ *then* $\widetilde{\mathbf{d}}(sk, c) = b$*; otherwise,* $\widetilde{\mathbf{d}}(sk, c) = \perp$.

Note that in the above definition if $(\mathbf{g}, \mathbf{e}, \mathbf{d})$ is a valid PKE scheme and $\mathsf{Q_s}$ satisfies the required condition then $(\widetilde{\mathbf{g}}, \mathbf{e}, \widetilde{\mathbf{d}})$ is also a valid PKE scheme. The resulting $\widetilde{\mathbf{g}}$, however, will not be injective if there are "collisions" between $\mathsf{Q_s}$ and $\mathbf{g}$. Nonetheless, the resulting $(\widetilde{\mathbf{g}}, \mathbf{e}, \widetilde{\mathbf{d}})$ is still both well-defined and valid.

We will use the following fact over and over again in the paper. Informally, it shows one particular situation where $\widetilde{\mathbf{d}}$ queries, defined as above, can be handled using full access to $(\mathbf{g}, \mathbf{e}, \mathbf{d})$ and partial access to $\mathbf{u}$.

Fact 1. *Let* $(\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{u}, \mathbf{w})$ *be a* Ψ*-valid oracle and let* $\mathcal{B}^{\mathbf{g},\mathbf{e},\mathbf{d},\mathbf{u},\mathbf{w}}(pk, \ldots)$ *be a CCA-valid adversary (Definition 4) with a challenge public key* pk*. (The set of* $\mathcal{B}$*'s challenge ciphertexts is not important for this discussion.) Let* $\mathsf{Q_s}$ *be a set of query/response pairs meeting the condition of Definition 5 and* $(\widetilde{\mathbf{g}}, \widetilde{\mathbf{d}}) = KeyImpose(\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathsf{Q_s})$*. Assuming* $(\langle \mathfrak{g}, * \rangle, pk) \notin \mathsf{Q_s}$*, then* $\mathcal{B}^{\mathbf{g},\mathbf{e},\mathbf{d},\mathbf{u},\mathbf{w}}(pk, \ldots)$*, by having* $\mathsf{Q_s}$ *as a separate input, can efficiently compute* $\widetilde{\mathbf{d}}(sk', c')$*, for all* sk' *and* c' *without violating the CCA condition.*

Proof. For any query $qu = \langle \widetilde{\mathbf{d}}, (sk', c') \rangle$, either (i) $(\langle \mathfrak{g}, sk' \rangle, *) \notin \mathsf{Q_s}$ or (ii) for some $pk' \neq pk$, $(\langle \mathfrak{g}, sk' \rangle, pk') \in \mathsf{Q_s}$. If (i) holds then $\widetilde{\mathbf{d}}(sk', c') = \mathbf{d}(sk', c')$ and so $\mathcal{B}$ can reply to qu by calling $\langle \mathbf{d}, (sk', c') \rangle$. If (ii) holds, the answer to qu can be determined by calling $\langle \mathbf{u}, (pk', c') \rangle$, which is a valid query for $\mathcal{B}$ as $pk' \neq pk$. □

We make the following two assumptions for any construction (G, E, D) discussed throughout.

Assumption 1. For any Ψ-valid $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d})$ we assume $G^{\mathbf{O}}$, $E^{\mathbf{O}}$ and $D^{\mathbf{O}}$, on inputs corresponding to security parameter n make exactly n^{ϑ} oracle calls (for $\vartheta \geq 1$) and that $G^{\mathbf{O}}(1^n)$ uses exactly n random bits.[2]

Assumption 2. We assume G, E and D, on inputs relative to security parameter 1^n only call their oracles under the same security parameter 1^n. This assumption is only made to simplify our exposition. Indeed, we are not aware of any construction that does not satisfy this assumption.

[2] Note that we do not claim that there exists a universal ϑ that works for all constructions $\mathcal{E} = (G, E, D)$. Rather, for any fixed construction (G, E, D) which we want to rule out (i.e., define a breaking oracle $\mathbf{T}$ for), we fix a ϑ that satisfies the stated conditions. Also, the assumption that G relative to any Ψ valid oracle uses n coins is not necessary; it can indeed be any fixed $p(n)$ number of coins, but assuming it to be n allows us to dispense with an additional parameter p.

We now describe our techniques for a simple class of constructions, those with oracle access of the form $(G^{\mathbf{g}}, E^{\mathbf{e}}, D^{\mathbf{d}})$. We first give the oracle $\mathbf{T}$, defined w.r.t. a fixed $(\mathbf{g}, \mathbf{e}, \mathbf{d})$ and a fixed construction (G, E, D), which helps us to break the seed-circular security of $(G^{\mathbf{g}}, E^{\mathbf{e}}, D^{\mathbf{d}})$. Fix (G, E, D) throughout this section, so we make the dependence of $\mathbf{T}$ on (G, E, D) implicit below. The oracle $\mathbf{T}$ is selected from a class of oracles, but it is convenient to define the output distribution of a randomly chosen $\mathbf{T}$ on an arbitrary given input, as we do below.

Description of T:
Oracles: $(\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{w})$
Input: $(1^n, PK, C_1, \ldots, C_n)$

1. Choose $(\mathbf{g}', S')$ uniformly at random from the set of all pairs satisfying (a) $\mathbf{g}'$ is Ψ-valid and (b) $G^{\mathbf{g}'}(1^n, S') = (*, PK)$. If no such a pair exists return $\perp$. Otherwise, let SK' be the secret key output by $G^{\mathbf{g}'}(1^n, S')$.
2. Let $\mathsf{Q_s}$ contain the symbolic versions of all query/response pairs made in the execution of $G^{\mathbf{g}'}(1^n, S')$. Define $(\widetilde{\mathbf{g}}, \widetilde{\mathbf{d}}) = KeyImpose(\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathsf{Q_s})$. Let QPub include any pk such that $\mathbf{w}(pk) = \top$ and $(\langle \mathfrak{g}, * \rangle, pk) \in \mathsf{Q_s}$.
3. Compute $S_{out} = D^{\widetilde{\mathbf{d}}}(SK', C_1 \ldots C_n)$. Execute $G^{\mathbf{g}}(S_{out})$ and if for all $pk \in \mathsf{QPub}$ the query/response $(\langle \mathbf{g}, * \rangle, pk)$ is made during the execution, then return S_{out}; otherwise, return $\perp$.

We now informally discuss why $\mathbf{T}$ provides the "desired" properties.

4.1.1 T Does Not Break CPA Security of a Random (g, e, d)

We show that any adversary $\mathcal{A}^{\mathbf{O},\mathbf{T}}(1^n, pk, c)$ against the CPA-security of $\mathbf{O}$ can be fully simulated without $\mathbf{T}$, by a CCA-valid adversary $\mathcal{B}^{\mathbf{O},\mathbf{u},\mathbf{w}}(1^n, pk, c)$ (See Definition 4). We then show any such $\mathcal{B}$ has a very small chance of breaking the security of $\mathbf{O}$, by relying on a special case of the following lemma which shows a random $\mathbf{O}$ is t-seed-circular secure in a strong sense. As notation whenever we write $f_1(n) \leq f_2(n)$ we mean that this holds asymptotically.

Lemma 1. *Let $t = t(n)$ be a poly. Let $\mathcal{B}$ be a CCA-valid oracle adversary (Definition 4), which has access to some Ψ-valid oracle $(\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{u}, \mathbf{w})$, and which makes at most $2^{n/4}$ queries and outputs a bit. It then holds that*

$$\Pr\left[\mathcal{B}^{\mathbf{O},\mathbf{u},\mathbf{w}}(1^n, pk_1, \ldots, pk_t, \mathbf{e}(pk_1, sk_2), \ldots, \mathbf{e}(pk_t, sk_1), \mathbf{e}(pk_1, b)) = b\right] \leq \frac{1}{2} + \frac{1}{2^{n/4}},$$

where $\mathcal{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{u}, \mathbf{w}) \leftarrow \Psi$, $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d})$, $b \leftarrow \{0,1\}$, $sk_i \leftarrow \{0,1\}^n$ and $pk_i = \mathbf{g}(sk_i)$ for $1 \leq i \leq t$.

Fix a Ψ-valid oracle $(\mathbf{O}, \mathbf{u}, \mathbf{w})$. We show that any adversary $\mathcal{A}^{\mathbf{O},\mathbf{T}}(1^n, pk, c)$, against the CPA-security of $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d})$, can be perfectly simulated by a CCA-valid adversary $\mathcal{B}^{\mathbf{O},\mathbf{u},\mathbf{w}}(1^n, pk, c)$ that makes a poly-related number of queries. The crux of our techniques lies in showing how $\mathcal{B}$, using $\mathbf{u}$ and $\mathbf{w}$, can simulate $\mathcal{A}$'s $\mathbf{T}$ access.

Specifically, $\mathcal{B}^{\mathbf{O},\mathbf{u},\mathbf{w}}(1^n, pk, c)$ starts running $\mathcal{A}^{\mathbf{O},\mathbf{T}}(1^n, pk, c)$ and forwards all $\mathcal{A}$'s $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d})$ queries to its own corresponding oracles.

To respond to a $\mathbf{T}$ query of the form $Tqu \stackrel{\text{def}}{=} \langle \mathbf{T}, (1^{n_1}, PK, C_1, \ldots, C_{n_1}) \rangle$ made by $\mathcal{A}$, $\mathcal{B}$ acts as follows (note it may be that $n_1 \neq n$, as $\mathcal{A}$ can make queries under different security parameters): $\mathcal{B}$ forms SK' and $\mathsf{Q_s}$ exactly as in Steps 1 and 2 of $\mathbf{T}$'s computation. It is able to do so since during these two computations no queries are made to the real oracles (though, a massive offline search is involved). Next, $\mathcal{B}$ starts simulating $D^{\widetilde{\mathbf{d}}}(SK', C_1, \ldots, C_{n_1})$, where $(\widetilde{\mathbf{g}}, \widetilde{\mathbf{d}}) = KeyImpose(\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathsf{Q_s})$. Since it is not clear how $\mathcal{B}$ can perform this decryption by only making a polynomial number of queries and without ever calling $\langle \mathbf{u}, (pk, c) \rangle$, we consider two possible cases:

(A) $(\langle \mathbf{g}, * \rangle, pk) \notin \mathsf{Q_s}$: In this case $\mathcal{B}$ can fully execute $D^{\widetilde{\mathbf{d}}}(SK', C_1, \ldots, C_{n_1})$, since by Fact 1 $\mathcal{B}$ can handle all encountered queries, which are all of type $\widetilde{\mathbf{d}}$. (Recall that pk is $\mathcal{B}$'s challenge public key.) Now if $S_{out} = D^{\widetilde{\mathbf{d}}}(SK', C_1, \ldots, C_{n_1})$, then $\mathcal{B}$ performs the rest of Step 3 of $\mathbf{T}$, which $\mathcal{B}$ can fully do since the rest only involves making $\mathbf{g}$ and $\mathbf{w}$ queries. Thus, $\mathcal{B}$ can find the answer to Tqu.

(B) $(\langle \mathbf{g}, * \rangle, pk) \in \mathsf{Q_s}$: In this case, recalling the definition of Qpub, we have $pk \in \mathsf{QPub}$, since pk is $\mathcal{B}$'s challenge public key and so by definition $\mathbf{w}(pk) = \top$. Thus, by the condition given in Step 3 of $\mathbf{T}$'s description, if $S_{out} = D^{\widetilde{\mathbf{d}}}(SK', C_1, \ldots, C_{n_1})$ then at least one of the following holds:

(a) The answer to Tqu is $\perp$; or

(b) The query/response pair $(\langle \mathbf{g}, * \rangle, pk)$ will show up during $G^{\mathbf{g}}(S_{out})$, i.e., $\mathbf{g}^{-1}(pk)$, $\mathcal{B}$'s challenge secret key, is revealed during $G^{\mathbf{g}}(S_{out})$.

We now claim that $\mathcal{B}$ *can find two strings* S_0 *and* S_1 *such that* $S_{out} \in \{S_0, S_1\}$. If this is the case, $\mathcal{B}$ can execute both $G^{\mathbf{g}}(S_0)$ and $G^{\mathbf{g}}(S_1)$; if during either execution a query/response $(\langle \mathbf{g}, * \rangle, pk)$ is observed, $\mathcal{B}$ has learned $\mathbf{g}^{-1}(pk)$, winning the game; otherwise, $\mathcal{B}$ in response to Tqu returns $\perp$, which is indeed the correct response.

It remains to demonstrate the claim. To find S_0 and S_1, $\mathcal{B}$ attempts to simulate $D^{\widetilde{\mathbf{d}}}(SK', C_1, \ldots, C_{n_1})$. For any query $qu = \langle \widetilde{\mathbf{d}}, (sk', c') \rangle$ encountered in the simulation, one of the following holds

(i) $(\langle \mathbf{g}, sk' \rangle, *) \notin \mathsf{Q_s}$; or

(ii) $(\langle \mathbf{g}, sk' \rangle, pk') \in \mathsf{Q_s}$ for $pk' \neq pk$; or

(iii) $(\langle \mathbf{g}, sk' \rangle, pk) \in \mathsf{Q_s}$ and $c' \neq c$; or

(iv) $(\langle \mathbf{g}, sk' \rangle, pk) \in \mathsf{Q_s}$ and $c' = c$.

$\mathcal{B}$ can find the answer to qu by querying $\langle \mathbf{d}, (sk', c') \rangle$ for Case (i), querying $\langle \mathbf{u}, (pk', c') \rangle$ for Case (ii), and querying $\langle \mathbf{u}, (pk, c') \rangle$ for Case (iii). The latter two are legitimate $\mathbf{u}$ queries (see Definition 4).

For Case (iv) $\mathcal{B}$ continues the execution of $D^{\widetilde{\mathbf{d}}}(SK', C_1, \ldots, C_{n_1})$ in two parallel branches BR_0 and BR_1, where $\mathcal{B}$ replies to qu with b on BR_b. On both branches $\mathcal{B}$ replies to queries for which Cases (i), (ii) and (iii) hold exactly as above. If on some branch $BR_{b'}$, still during the execution of $D^{\widetilde{\mathbf{d}}}(SK', C_1, \ldots, C_{n_1})$, for a query qu' Case (iv) holds again (i.e.,

$qu' = \langle \widetilde{\mathbf{d}}, (sk', c)\rangle$ and $(\langle \mathfrak{g}, sk'\rangle, pk) \in \mathsf{Q_s}$) $\mathcal{B}$ replies to qu' with b', making it consistent with the previous reply. Thus, these two branches result in two strings S_0, S_1 satisfying the claim.

By invoking Lemma 1 we deduce that for random $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d})$ and $\mathbf{T}$, any $\mathcal{A}^{\mathbf{O},\mathbf{T}}(pk, c)$ that makes at most, say, $2^{n/5}$ queries (basically any number m of queries where $2^{n/4}/m$ is super-polynomial) has advantage at most $\frac{1}{2} + \frac{1}{2^{n/4}}$ of computing b, where $sk \leftarrow \{0,1\}^n$, $pk = \mathbf{g}(sk)$ and $c \leftarrow \mathbf{e}(pk, b)$.

4.1.2 T Breaks Seed-Circular Security of (G, E, D)

We show

CLAIM A. $\mathbf{T}$ is useful if used honestly: For $\mathcal{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{u}, \mathbf{w}) \leftarrow \Psi$, $S \leftarrow \{0,1\}^n$, $(SK, PK) = G^{\mathbf{g}}(S)$ and $(C_1, \ldots, C_n) \leftarrow E^{\mathbf{e}}(PK, S)$, the probability that $\mathbf{T}(1^n, PK, C_1, \ldots, C_n)$ does not return S is exponentially small.

To prove CLAIM A, we need the following simple information-theoretic lemma, showing that the probability that an adversary can "forge" a public key is small.

Lemma 2. *Let $\mathcal{B}$ be an oracle adversary, which has access to some Ψ-valid oracle $\mathcal{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{u}, \mathbf{w})$, and which on input 1^n makes a list Que of at most 2^n queries and outputs a public key $pk_{out} \in \{0,1\}^{5n}$. It then holds that*

$$\Pr_{\mathcal{O} \leftarrow \Psi}[\mathbf{w}(pk_{out}) = \top \text{ and } (\langle \mathbf{g}, *\rangle, pk_{out}) \notin \mathsf{Que}] \leq \frac{1}{2^{2n}}.$$

Proof of CLAIM A. *Let the variables $\mathbf{g}'$, S', $\mathsf{Q_s}$, SK', $\widetilde{\mathbf{g}}$, $\widetilde{\mathbf{d}}$ and S_{out} be sampled as in $\mathbf{T}(1^n, PK, C_1, \ldots, C_n)$. Recall that $(SK', PK) = G^{\mathsf{Q_s}}(1^n, S')$, that $(\widetilde{\mathbf{g}}, \widetilde{\mathbf{d}}) = KeyImpose(\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathsf{Q_s})$ and that $S_{out} = D^{\widetilde{\mathbf{d}}}(SK', C_1 \ldots C_n)$.*

Recall the way $S, PK, C_1, \ldots, C_n$ are chosen in the claim. We first claim $S_{out} = S$. This follows since (a) $(C_1, \ldots, C_n) \leftarrow E^{\mathbf{e}}(PK, S)$, (b) $(\widetilde{\mathbf{g}}, \mathbf{e}, \widetilde{\mathbf{d}})$ is a correct PKE, and (c) $(SK', PK) = G^{\widetilde{\mathbf{g}}}(1^n; S')$, since $\widetilde{\mathbf{g}}$ agrees with $\mathsf{Q_s}$. Thus, by the correctness of (G, E, D), $S_{out} = S$. Thus, $\mathbf{T}(1^n, PK, C_1, \ldots, C_n)$ either returns S or $\perp$.

Let Fail be the event $\mathbf{T}(PK, C_1, \ldots, C_n) = \perp$. We show how to successfully forge a public $pk \in \{0,1\}^{5n}$ whenever Fail holds. By Lemma 2 we will then have $\Pr[Fail] \leq \frac{1}{2^{2n}}$, implying that with probability at least $1 - \frac{1}{2^{2n}}$, $\mathbf{T}(PK, C_1, \ldots, C_n)$ returns S. We first start with some intuition behind the forgery.

Recall that $S_{out} = S$. By definition, Fail occurs if there exists pk s.t. (a) $\mathbf{w}(pk) = \top$, (b) pk is embedded in $\mathsf{Q_s}$ (i.e., $(\langle \mathfrak{g}, *\rangle, pk) \in \mathsf{Q_s}$) and (c) the query/response $(\langle \mathbf{g}, *\rangle, pk)$ does not show up during $G^{\mathbf{g}}(1^n, S)$. Now the forgery is enabled by the fact that $\mathsf{Q_s}$ is produced based on PK in offline mode. (Recall that $(SK, PK) = G^{\mathbf{g}}(1^n, S)$). The only thing left is to prove that the forged pk is indeed in $\{0,1\}^{5n}$. The reason is the following: since $(\langle \mathfrak{g}, *\rangle, pk) \in \mathsf{Q_s}$, the public key pk shows up as the response to a $\mathbf{g}'$ query during $G^{\mathbf{g}'}(1^n, S')$. Recalling that $\mathbf{g}'$ is Ψ-valid (see Step 1 of $\mathbf{T}$'s computation), by Assumption 2 we have $pk \in \{0,1\}^{5n}$. Given this intuition, the forging adversary $\mathcal{A}$ works as follows.

$\mathcal{A}^{\mathcal{O}}(1^n)$ generates $S \leftarrow \{0,1\}^n$ and $(SK, PK) = G^{\mathbf{g}}(S)$; it then samples a Ψ-valid function $\mathbf{g}'$ and a seed S' in such a way that $G^{\mathbf{g}'}(1^n, S') = (*, PK)$ and lets $\mathsf{Q_s}$ contain the symbolic versions of all query/response pairs made to $\mathbf{g}'$. Denoting by Que the set of all query/response pairs of $\mathcal{A}$ so far (which was populated only during $G^{\mathbf{g}}(S)$), for all pk s.t. $(\langle \mathfrak{g}, * \rangle, pk) \in \mathsf{Q_s}$ and $(\langle \mathbf{g}, * \rangle, pk) \notin \mathsf{Que}$, $\mathcal{A}$ calls $\langle \mathbf{w}, pk \rangle$: as soon as $\mathcal{A}$ receives $\top$ in response, it returns pk. □

We can now, using standard techniques, combine the two facts above about $\mathbf{T}$ to rule out fully-blackbox reductions for the construction type considered.

We conclude this subsection with a remark. The separation proved in this subsection will hold even if the candidate construction $\mathcal{E}$ is full length. This can easily be checked, considering nowhere in our analysis do we use the fact that E is a single-bit encryption algorithm. This may briefly be thought of as contradicting the positive construction basing full-length 1-seed circular security on CPA security! However, the catch here is that the positive construction alluded to earlier does not belong in the class of constructions ruled out here, since the constructed E calls the base key-generation algorithm. When discussing the general separation result in Sect. 5 we will point out exactly where our separation fails if the constructed scheme is full-length.

4.2 Bit t-seed-circular Security $\not\Rightarrow$ Full-Length $(t+1)$-seed-circular Security

For simplicity, we show how to separate full-length 2-seed-circular security from bit 1-seed circular security, as this case already captures most of the underlying techniques. Since in this case the seed in the constructed scheme is encrypted as a whole we denote a seed encryption as $C \leftarrow E_{PK}(S)$. Fix the proposed full-length encryption construction (G, E, D), for which we will define a weakening oracle $\mathbf{T_2}$ in such a way that $\mathbf{T_2}$ breaks the 2-seed circular security of $(G^{\mathbf{g}}, E^{\mathbf{e}}, D^{\mathbf{d}})$, but not the 1-seed circular security of $(\mathbf{g}, \mathbf{e}, \mathbf{d})$.

For this new setting, we cannot use the previous approach, mainly because the analysis there for showing that $\mathbf{T}$ is simulatable by a CCA-valid adversary against $(\mathbf{g}, \mathbf{e}, \mathbf{d})$ (Subsect. 4.1.1) heavily relies on the fact that only one challenge ciphertext is present, whose value can be guessed in two branches. That simulation trick will fail here because an adversary against the bit 1-seed circular security of $(\mathbf{g}, \mathbf{e}, \mathbf{d})$, which will have access to $\mathbf{T_2}$ and which we want to simulate without $\mathbf{T_2}$, is provided with $n+1$ ciphertexts. Thus, we need some new ideas for the oracle $\mathbf{T_2}$, outlined below. We also use some of the previous ideas.

$\mathbf{T_2}$ accepts inputs of the form (PK_0, PK_1, C_0, C_1), where purportedly C_i, for $i = 0, 1$, is the encryption of PK_{1-i}'s seed under $E(PK_i, \cdot)$. Intuitively, $\mathbf{T_2}$ will decrypts C_0 and C_1 relative to, respectively, oracles $\widetilde{\mathbf{O_0}}$ and $\widetilde{\mathbf{O_1}}$, obtained by superimposing two sampled sets $\mathsf{Q_s^0}$ and $\mathsf{Q_s^1}$, meeting a certain condition, on $\mathbf{O}$. Specifically, $\mathbf{T_2}$ samples two sets of query/response pairs $\mathsf{Q_s^0}$ and $\mathsf{Q_s^1}$ in such a way that for $i = 0, 1$ (a) $G^{\mathsf{Q_s^i}} = (SK_i', PK_i)$ for some SK_i' and (b) the sets of embedded public keys in $\mathsf{Q_s^0}$ and $\mathsf{Q_s^1}$ are disjoint, namely for all pk: if $(\langle \mathfrak{g}, * \rangle, pk) \in \mathsf{Q_s^0}$ then $(\langle \mathfrak{g}, * \rangle, pk) \notin \mathsf{Q_s^1}$. (If such $\mathsf{Q_s^0}$ and $\mathsf{Q_s^1}$ cannot be found,

$\mathbf{T_2}$ returns $\perp$.) Then, $\mathbf{T_2}$ forms $(\widetilde{\mathbf{g}}_i, \widetilde{\mathbf{d}}_i) = KeyImpose(\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathsf{Q}^{\mathsf{i}}_{\mathsf{s}})$, and $S^0_{out} = D^{\widetilde{\mathbf{d}}_1}(SK'_1, C_1)$ and $S^1_{out} = D^{\widetilde{\mathbf{d}}_0}(SK'_0, C_0)$. Finally, $\mathbf{T_2}$ returns S^0_{out} if for both $i = 0, 1$ all embedded public keys in $\mathsf{Q}^{\mathsf{i}}_{\mathsf{s}}$ appear during the execution of $G^{\mathbf{g}}(S^i_{out})$.

The check (b) above is aimed at making $\mathbf{T_2}$ simulatable using $\mathbf{u}$ and $\mathbf{w}$ oracles: namely, to make any 1-seed circular security adversary $\mathcal{A}^{\mathbf{O},\mathbf{T_2}}(pk, c_1, \ldots, c_n, c)$ against $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d})$ simulatable by CCA-valid adversary $\mathcal{B}^{\mathbf{O},\mathbf{u},\mathbf{w}}(pk, c_1, \ldots, c_n, c)$. The main idea behind the simulation is that, for any query $\langle \mathbf{T_2}, (PK_0, PK_1, C_0, C_1)\rangle$ of $\mathcal{A}$, the adversary $\mathcal{B}$ will be able to decrypt at least one of C_i's, specifically the one for which $(\langle \mathfrak{g}, * \rangle, pk) \notin \mathsf{Q}^{\mathsf{i}}_{\mathsf{s}}$. This follows by Fact 1. (Recall pk is $\mathcal{B}$'s challenge public key.) If for the other index (i.e., $1-i$) it holds that $(\langle \mathfrak{g}, * \rangle, pk) \in \mathsf{Q}^{1-\mathsf{i}}_{\mathsf{s}}$ then as before we can show that either the answer to the underlying $\mathbf{T_2}$ query is $\perp$, or $\mathcal{B}$ will learn its challenge secret key (i.e., $\mathbf{g}^{-1}(pk)$) along the way.

The check (b) however may make the oracle $\mathbf{T_2}$ too weak to break the 2-seed circular security of $(G^{\mathbf{O}}, E^{\mathbf{O}}, D^{\mathbf{O}})$. In particular, if there are pk's that occur w.h.p. as responses to $\mathbf{g}$ queries during a random execution of $G^{\mathbf{g}}(1^n)$, then $\mathbf{T_2}$, even on "honest" inputs, may return $\perp$ too often. To resolve this problem, we first sample a large number of executions of $G^{\mathbf{O}}$, record all the query/response pairs and make $\mathsf{Q}^0_{\mathsf{s}}$ and $\mathsf{Q}^1_{\mathsf{s}}$ be consistent with this information.

We now describe the oracle $\mathbf{T_2}$.

Description of $\mathbf{T_2}$:
Oracles: $(\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{w})$
Input: $(1^n, PK_0, PK_1, C_0, C_1)$

1. **Learning heavy key-generation queries:** Execute $G^{\mathbf{g}}(1^n)$ ϱ times independently at random and record all query/response pairs to Freq. (We instantiate ϱ later.) For any $(\langle \mathbf{g}, * \rangle, pk) \in \mathsf{Freq}$ add pk to $\mathsf{FreqPub}$.
2. **Sampling oracles/secret keys consistent with Freq, PK_1 and PK_2.** For $i = 0, 1$:
   - choose $(\mathbf{g}'_i, S'_i)$ uniformly at random from the set of all pairs satisfying (a) $\mathbf{g}'_i$ is Ψ-valid and is consistent with Freq and (b) $G^{\mathbf{g}'_i}(1^n, S'_i) = (*, PK_i)$. (If no such a pair exists return $\perp$.) Let SK'_i be the secret key output by $G^{\mathbf{g}'_i}(1^n, S'_i)$.
   - Let $\mathsf{Q}^{\mathsf{i}}_{\mathsf{s}}$ contain the symbolic versions of all query/response pairs made in the execution of $G^{\mathbf{g}'_i}(1^n, S'_i)$. Define $(\widetilde{\mathbf{g}}_{\mathbf{i}}, \widetilde{\mathbf{d}}_{\mathbf{i}}) = KeyImpose(\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathsf{Q}^{\mathsf{i}}_{\mathsf{s}})$. Let $\mathsf{QPub}_{\mathsf{i}}$ have any pk s.t. $\mathbf{w}(pk) = \top$ and $(\langle \mathfrak{g}, * \rangle, pk) \in \mathsf{Q}^{\mathsf{i}}_{\mathsf{s}}$.
3. If $(\mathsf{QPub}_0 \cap \mathsf{QPub}_1) \setminus \mathsf{FreqPub} \neq \emptyset$ then halt and return $\perp$.
4. Compute $S^1_{out} = D^{\widetilde{\mathbf{d}}_0}(SK'_0, C_0)$ and $S^0_{out} = D^{\widetilde{\mathbf{d}}_1}(SK'_1, C_1)$. Return S^0_{out} if the following condition holds for both $i = 0, 1$, and return $\perp$, otherwise: For all $pk \in \mathsf{QPub}_{\mathsf{i}} \setminus \mathsf{FreqPub}$ the query/response $(\langle \mathbf{g}, * \rangle, pk)$ is made during the execution of $G^{\mathbf{g}}(S^i_{out})$.

$\mathbf{T_2}$ does not break 1-seed-circular security of $\mathbf{O}$. We show any adversary $\mathcal{A}^{\mathbf{O},\mathbf{T_2}}(1^n, pk, c_1, \ldots, c_n, c)$, against 1-seed-circular-security of $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d})$ can

be simulated by a CCA-valid adversary $\mathcal{B}^{\mathbf{O},\mathbf{u},\mathbf{w}}(1^n, pk, c_1, \ldots, c_n, c)$ that makes a poly-related number of queries. By Lemma 1 we then obtain our desired result.

The main challenge for $\mathcal{B}$ is to handle $\mathcal{A}$'s $\mathbf{T_2}$ queries. Fix a $\mathbf{T_2}$ query $Tqu = \langle \mathbf{T_2}, (1^{n_1}, PK, C_1, C_2)\rangle$ of $\mathcal{A}$. To reply to Tqu, $\mathcal{B}$ forms $\mathsf{FreqPub}$, Q^0_s, Q^1_s, SK'_0 and SK'_1 as in $\mathbf{T_2}$'s computation, which $\mathbf{B}$ can perfectly do. Without loss of generality assume $pk \notin \mathsf{FreqPub}$, since otherwise $\mathcal{B}$ has found its challenge secret key. Also, assume for some $i \in \{0,1\}$ $pk \notin \mathsf{QPub}_\mathsf{i}$ because otherwise by Line 3 the answer to Tqu is $\perp$. In what follows assume $pk \notin \mathsf{QPub}_1$. (The same argument goes through if $pk \notin \mathsf{QPub}_0$.)

$\mathcal{B}$ forms $S^0_{out} = D^{\widetilde{\mathbf{d}_1}}(SK'_1, C_1)$, where $(\widetilde{\mathbf{g}_1}, \widetilde{\mathbf{d}_1}) = KeyImpose(\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathsf{Q}^1_\mathsf{s})$. By Fact 1, $\mathcal{B}$ is perfectly able to run this decryption. Now consider two cases:

1. $pk \notin \mathsf{QPub}_0$: in this case again $\mathcal{B}$ can compute $S^1_{out} = D^{\widetilde{\mathbf{d}_0}}(SK'_0, C_0)$, where $(\widetilde{\mathbf{g}_0}, \widetilde{\mathbf{d}_0}) = KeyImpose(\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathsf{Q}^0_\mathsf{s})$. Having both S^0_{out} and S^1_{out} $\mathcal{B}$ can easily perform the rest of $\mathbf{T_2}$'s computation (which only involves $\mathbf{g}$ queries).
2. $pk \in \mathsf{QPub}_0$: in this case by Line 4 of $\mathbf{T_2}$'s computation, either the answer to Tqu is $\perp$ or pk's corresponding secret key turns up during the execution of $G^{\mathbf{O}}(S^0_{out})$. (Recall that $pk \notin \mathsf{FreqPub}$.) Thus, $\mathbf{B}$ either finds its challenge secret key or finds out that the answer to Tqu is $\perp$.

$\mathbf{T_2}$ breaks 2-seed-circular security: For $\mathcal{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{u}, \mathbf{w}) \leftarrow \Psi$, $S_i \leftarrow \{0,1\}^n$, $(SK_i, PK_i) = G^{\mathbf{g}}(S_i)$ for $i = 0, 1$, and $C_1 \leftarrow E^{\mathbf{e}}(PK_1, S_0)$ and $C_0 \leftarrow E^{\mathbf{e}}(PK_0, S_1)$ we show the probability that $\mathbf{T_2}(1^n, PK_0, PK_1, C_0, C_1)$ does not return S_0 is exponentially small. First, as in the corresponding proof in Subsect. 4.1 we can easily show it is always the case that $S_i = S^i_{out}$ for $i = 0, 1$. Thus, the probability that $\mathbf{T_2}(1^n, PK_0, PK_1, C_0, C_1)$ does not return S_0 is the probability that one of the bad events in Lines 3 and 4 of $\mathbf{T_2}$'s computation holds. Let Ev be the event that $\mathbf{T_2}(1^n, PK_0, PK_1, C_0, C_1)$ does not return S_0.

The bad events in Lines 3 and 4 correspond to events $Ev1$ and $Ev2$, defined as follows: $Ev1 = (\mathsf{QPub}_0 \cap \mathsf{QPub}_1) \setminus \mathsf{FreqPub} \neq \emptyset$ and

$$Ev2 = ((\mathsf{QPub}_0 \not\subseteq \mathsf{RealPub}_0 \cup \mathsf{FreqPub}) \vee (\mathsf{QPub}_1 \not\subseteq \mathsf{RealPub}_1 \cup \mathsf{FreqPub})), \quad (2)$$

where $\mathsf{RealPub}_\mathsf{i} = \{pk \mid$ the query/response $(\langle \mathbf{g}, * \rangle, pk)$ occurs during $G^{\mathbf{g}}(S_i)\}$. Note that for $Ev2$ we use the fact that $S_i = S^i_{out}$. We have $\Pr[Ev] \leq \Pr[Ev2] + \Pr[Ev1 \wedge \overline{Ev2}]$.

First, using the same technique as in Subsect. 4.1.2 we can show $\Pr[Ev2]$ is exponentially small. To bound $\Pr[Ev1 \wedge \overline{Ev2}]$, note whenever $Ev1 \wedge \overline{Ev2}$ happens, the event $Ev3$, defined below, happens:

$$Ev3 = (\mathsf{RealPub}_0 \cap \mathsf{RealPub}_1) \setminus \mathsf{FreqPub} \neq \emptyset.$$

Thus, we show how to bound the probability of $Ev3$. That is, the probability that there exists pk such that $pk \in \mathsf{RealPub}_0 \cap \mathsf{RealPub}_1$, but $pk \notin \mathsf{FreqPub}$. Intuitively, this probability should be small because if $pk \in \mathsf{RealPub}_0 \cap \mathsf{RealPub}_1$—namely, the query/response $(\langle \mathbf{g}, * \rangle, pk)$ occurs during both $G^{\mathbf{g}}(S_0)$ and $G^{\mathbf{g}}(S_1)$—then $(\langle \mathbf{g}, * \rangle, pk)$ should also occur at least once during the many random executions of $G^{\mathbf{g}}(1^n)$ performed in Step 1 of $\mathbf{T_2}$'s computation, and thus it should

be that $pk \in \mathsf{FreqPub}$. Using this line of reasoning we can use Chernoff Bounds to upperbound the probability of $Ev3$ by any arbitrary inverse-polynomial, by instantiating the value of ϱ (Step 1 of $\mathbf{T_2}$'s computation) accordingly.

5 CPA Security $\not\Rightarrow$ Bit 1-seed-circular Security: General

In this section we describe the oracle $\mathbf{T}$ for general bit-encryption constructions of 1-seed circular security, and in the following two sections we prove that this oracle provides a separation.

Intuition. As in the previous section the main idea is to have $\mathbf{T}$, on input $(PK, C_1, \ldots, C_n)$, decrypt $C_1, \ldots, C_n$ relative to some $\widetilde{\mathbf{O}} = (\widetilde{\mathbf{g}}, \widetilde{\mathbf{e}}, \widetilde{\mathbf{d}})$, satisfying (a) $G^{\widetilde{\mathbf{O}}}$ produces $(*, PK)$ and (b) for any b with high probability $E^{\mathbf{O}}(PK, b, R) = E^{\widetilde{\mathbf{O}}}(PK, b, R)$. To obtain $\widetilde{\mathbf{O}}$ we may be tempted to proceed exactly as before, by sampling a set of query/response pairs Q and a seed S' such that $G^{\mathsf{Q}}(S') = (*, PK)$ and then *superimposing* Q (which now has all types of queries) on $\mathbf{O}$. While the resulting $\widetilde{\mathbf{O}}$ satisfies Condition (a) it is not clear if Condition (b) is satisfied: The problem is there may be queries q asked quite frequently during random executions of $E^{\mathbf{O}}(PK, b)$ (call them *heavy*), and which may also occur in Q and receive a different response there. To overcome this problem we first run $E^{\mathbf{O}}(PK, b)$ for $b = 0, 1$ many times and collect all observed query/response pairs in a set Freq. (This is formalized in Definition 6.) We then force the sampled set Q to be *consistent* with Freq. Finally, we show how to superimpose Q on $\mathbf{O}$ to obtain $\widetilde{\mathbf{O}}$.

Setting things up. Fix the proposed construction (G, E, D). We now give an assumption to make our analysis easier and then give definitions formalizing the steps sketched above. We then use these definitions to define the oracle $\mathbf{T}$.

Assumption 3. We assume any oracle algorithm that has access to both $\mathbf{g}$ and $\mathbf{d}$ always queries $\langle \mathbf{g}, sk \rangle$ before querying $\langle \mathbf{d}, (sk, *) \rangle$. Also, we assume w.l.o.g. that G never calls the decryption algorithm of the base scheme, $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d})$. (For $\langle \mathbf{d}, (sk, c) \rangle$: letting $pk = \mathbf{g}(sk)$, either the query/response $(\langle \mathbf{e}, (pk, *, *) \rangle, c)$ was already made in which case G knows the answer, or the answer w.h.p. is $\perp$.) For ease of notation we keep $\mathbf{d}$ as a superscript to G and write $G^{\mathbf{O}}$.

Definition 6. *We define the following probabilistic procedure, FreqQue.*

- ***Oracles:*** $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{u})$
- ***Input:*** *A security parameter* 1^n *(left implicit), public key* PK *and* $p \in \mathbb{N}$.
- ***Output:*** *A set* Freq *formed as follows. For both* $b = 0, 1$ *run* $E^{\mathbf{O}}(1^n, PK, b)$ *independently* p *times and add the symbolic versions of all query/response pairs to* Freq. *Moreover, for any* $(\langle \mathfrak{d}, (sk, c) \rangle, *) \in \mathsf{Freq}$ *if* $\mathbf{u}(\mathbf{g}(sk), c) = (b', r') \neq \perp$ *add* $(\langle \mathfrak{e}, (pk, b', r') \rangle, c)$ *to* Freq.
  Note that by Assumption 1 $|\mathsf{Freq}| \leq 2pn^{\vartheta} + 2pn^{\vartheta} = 4pn^{\vartheta}$.

In the above definition apart from the actual observed query/response pairs we also enhanced Freq with some pairs obtained based on $(\langle \mathfrak{d}, (sk, c)\rangle, *)$ query/response pairs. This enhancement is only made to make some of the proofs simpler.

We say that oracle $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d})$ is *consistent* with (or *agrees with*) a symbolic query/response pair $(\langle \mathfrak{g}, sk\rangle, pk)$ if $\mathbf{g}(sk) = pk$. The same definition can be given for other types of query/response pairs. We say $\mathbf{O}$ is consistent with a set of query/response pairs if $\mathbf{O}$ agrees with each element in the set.

Definition 7. *We define the following procedure we call ConsOrc.*

- ***Input:*** *a public key PK and a set Freq of symbolic query/response pairs.*
- ***Output:*** *a secret key SK' and query/response sets $\mathsf{Q_s}$, $\mathsf{Q_c}$ sampled as follows.*
  - *Sample $(\mathbf{g}', \mathbf{e}', \mathbf{d}', S')$ uniformly at random under the constraints that $\mathbf{O}' = (\mathbf{g}', \mathbf{e}', \mathbf{d}')$ is Ψ-valid and is consistent with Freq and that $G^{\mathbf{O}'}(S') = (*, PK)$. If no such a tuple exists, return $\perp$.*
  - *Let SK' be the secret-key outputted by $G^{\mathbf{O}'}(S')$ and let the sets $\mathsf{Q_s}$ and $\mathsf{Q_c}$ contain, respectively, the symbolic versions of all query/response pairs made to $\mathbf{g}'$ and $\mathbf{e}'$. (Recall by Assumption 3 no $\mathbf{d}'$-query is made.)*

In Definition 5 we defined the task of superimposing a set of $\mathfrak{g}$ type query/response pairs on an oracle $(\mathbf{g}, \mathbf{e}, \mathbf{d})$. We now define the task of superimposing a set $\mathsf{Q_c}$ of $\mathfrak{e}$ queries on $(\mathbf{g}, \mathbf{e}, \mathbf{d})$: the result will be $(\mathbf{e}_{imp}, \mathbf{d}_{imp})$, perturbed versions of $(\mathbf{e}, \mathbf{d})$. Intuitively, we want $(\mathbf{g}, \mathbf{e}_{imp}, \mathbf{d}_{imp})$ to form a PKE, $\mathbf{e}_{imp}$ to agree with $\mathsf{Q_c}$ and $(\mathbf{e}_{imp}, \mathbf{d}_{imp})$ to agree as much as possible with $(\mathbf{e}, \mathbf{d})$.

Definition 8. *We define the following procedure we call EncImpose.*

- ***Input:*** *a Ψ-valid $(\mathbf{g}, \mathbf{e}, \mathbf{d})$ and a set*

$$\mathsf{Q_c} = \{(\langle \mathfrak{e}, (pk_1, b_1, r_1)\rangle, c_1), \ldots, (\langle \mathfrak{e}, (pk_p, b_p, r_p)\rangle, c_p)\},$$

  Note that pk_i's above are not necessarily distinct.
- ***Output:*** *$(\mathbf{e}_{imp}, \mathbf{d}_{imp})$, defined as follows. First, let $\mathsf{W} = \{(pk_1, c_1), \ldots, (pk_p, c_p)\}$ and $\mathsf{W}' = \{(pk_1, \mathbf{e}(pk_1, b_1, r_1)), \ldots, (pk_p, \mathbf{e}(pk_p, b_p, r_p))\}$. Define*

$$\mathbf{e}_{imp}(pk, b, r) = \begin{cases} c_i & \text{if } (pk, b, r) = (pk_i, b_i, r_i), \text{ for some } 1 \le i \le p \\ \hat{c} & \text{if } (pk, \mathbf{e}(pk, b, r)) \in \mathsf{W} \\ \mathbf{e}(pk, b, r) & \text{otherwise} \end{cases} \tag{3}$$

  where $\hat{c}$ is defined as follows: Letting x be the smallest integer such that $(pk, \mathbf{e}(pk, b, r + x)) \notin \mathsf{W} \cup \mathsf{W}'$ we set $\hat{c} = \mathbf{e}(pk, b, r + x)$. Here, $r + x$ is done using a standard method.

$$\mathbf{d}_{imp}(sk, c) = \begin{cases} b_i & \text{if } \mathbf{g}(sk) = pk_i \text{ and } c = c_i \text{ for some } 1 \le i \le p \\ \mathbf{d}(sk, c) & \text{otherwise} \end{cases} \tag{4}$$

We justify the second case of $\mathbf{e}_{imp}$'s definition: if $(pk, \mathbf{e}(pk, b, r)) \in \mathsf{W}$, say $(pk, \mathbf{e}(pk, b, r)) = (pk_i, c_i)$, we cannot set $\mathbf{e}_{imp}(pk, b, r) = \mathbf{e}(pk, b, r)$ as we have already set $c_i = \mathbf{e}_{imp}(pk_i, b_i, r_i)$: in particular, $\mathbf{e}_{imp}$ will be rendered incorrect if $b_i \neq b$. Thus, we keep shifting $\mathbf{e}(pk, b, r)$ (by adding x to r) until we hit a ciphertext $\hat{c}$ s.t. $(pk, \hat{c}) \notin \mathsf{W} \cup \mathsf{W}'$. The requirement $(pk, \hat{c}) \notin \mathsf{W}'$ is stronger than necessary, but will simplify some proofs. Note $\mathbf{e}_{imp}$ is not necessarily injective.

Description of T. We define the oracle $\mathbf{T}$. We first describe the output distribution of a random $\mathbf{T}$ on a single input-call, $(1^n, PK, C_1, \ldots, C_n)$, and then describe the underlying distribution from which $\mathbf{T}$ is chosen.
Oracles: $\mathcal{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{u}, \mathbf{w})$. Denote $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d})$.
Input: $(1^n, PK, C_1, \ldots, C_n)$ **Operations:**

1. **Learning frequent queries:** Let $\mathsf{Freq} \leftarrow FreqQue^{\mathbf{O},\mathbf{u}}(PK, n^{23\vartheta})$. Define $\mathsf{FreqPub}$ to be the set of public keys pk such that $(\langle \mathfrak{g}, * \rangle, pk) \in \mathsf{Freq}$.
2. **Sampling oracle/secret-key consistent with PK and Freq:** Sample
$$(SK', \mathsf{Q_s}, \mathsf{Q_c}) \leftarrow ConsOrc(PK, \mathsf{Freq}). \tag{5}$$
3. **Defining intermediate oracles:** Define
$$(\mathbf{e}_{imp}, \mathbf{d}_{imp}) = EncImpose(\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathsf{Q_c})$$
$$(\widetilde{\mathbf{g}}, \widetilde{\mathbf{d}}) = KeyImpose(\mathbf{g}, \mathbf{e}_{imp}, \mathbf{d}_{imp}, \mathsf{Q_s}).$$
Let $\widetilde{\mathbf{e}} = \mathbf{e}_{imp}$, and $\widetilde{\mathbf{O}} = (\widetilde{\mathbf{g}}, \widetilde{\mathbf{e}}, \widetilde{\mathbf{d}})$. Let QPub contain any pk such that $\mathbf{w}(pk) = \top$ and $(\langle \mathfrak{g}, * \rangle, pk) \in \mathsf{Q_s}$.
4. **Decrypting the encrypted input:** Compute $S_{out} = D^{\widetilde{\mathbf{O}}}(SK', C_1 \ldots C_n)$.
5. **Returning S_{out} subject to a check:** Run $G^{\mathbf{O}}(S_{out})$ and let $\mathsf{EmbedPub}$ contain any pk such that the query/response $(\langle \mathbf{g}, * \rangle, pk)$ is made during $G^{\mathbf{O}}(S_{out})$. If $\mathsf{QPub} \subseteq \mathsf{EmbedPub} \cup \mathsf{FreqPub}$ return S_{out}; else, return $\perp$.

Notation. $Tvars^{\mathcal{O}}(PK)$ denotes the random variable $(\mathsf{Freq}, SK', \mathsf{Q_s}, \mathsf{Q_c}, \widetilde{\mathbf{O}})$ obtained in the execution of $\mathbf{T}$ above w.r.t. $\mathcal{O}$ and PK. Note none of these random variables depend on $(C_1, \ldots, C_n)$. For the reader's convenience, we provide a table summary of how all these variables sampled in the last page of the paper.

Remark about T. Note that the only part of the oracle $\mathbf{T}$ that involves making random choices are Step 1 (sampling from $FreqQue^{\mathbf{O},\mathbf{u}}(PK, n^{23\vartheta})$) and Step 2 (sampling from $ConsOrc(PK, \mathsf{Freq})$). The number of random coins required to do the sampling in Step 1 is obviously finite. For Step 2 recall that the output of $ConsOrc(PK, \mathsf{Freq})$ is formed based on sampling a Ψ-valid random oracle $\mathbf{O}'$ that is consistent with Freq and also that $G^{\mathbf{O}'}(1^n)$ generates PK (based on some seed). By default, $\mathbf{O}'$ should be defined for all security parameters. However, by Assumption 2 it suffices to sample $\mathbf{O}'$ only for security parameters n. Thus, for any fixed input $(1^n, PK, C_1, \ldots, C_n)$, the amount of randomness used by a random $\mathbf{T}$ to compute $\mathbf{T}(1^n, PK, C_1, \ldots, C_n)$ is finite.

Sampling space of T. We now explain how to choose a random $\mathbf{T}$. In particular, we would like a randomly chosen $\mathbf{T}$, if queried under a single input many times, to always return the same output. To this end, every possible $\mathbf{T}$ comes with a collection of random-coin strings, where for every possible query $qu = (1^n, PK, C_1, \ldots, C_n)$ to $\mathbf{T}$, the collection has a corresponding random-coin string $Coin_{qu}$, used by $\mathbf{T}$ to make the random choices that appear during the computation of $\mathbf{T}(qu)$. When we write $\Pr_{\mathbf{T}}[]$ we mean the probability is computed over a $\mathbf{T}$ chosen uniformly at random from the above-mentioned space.

5.1 T Breaks 1-seed-circular Security of (G, E, D)

We show if $\mathbf{T}$ is called honestly (i.e., on a random public key and a random encryption of the underlying seed) it will return the seed with high probability. To formalize the statement we define the following *environment* that specifies a random choice of $(\mathbf{O}, \mathbf{u}, \mathbf{w})$ plus those underlying an honest random input to $\mathbf{T}$.

Environment: $Env(n)$: Output $(\mathbf{O}, \mathbf{u}, \mathbf{w}, S, PK, C_1, \ldots, C_n)$, where:

1. $(\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{u}, \mathbf{w}) \leftarrow \Psi$ and $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d})$;
2. $S \leftarrow \{0,1\}^n$, $(SK, PK) \leftarrow G^{\mathbf{O}}(S)$ and $(C_1, \ldots, C_n) \leftarrow E^{\mathbf{O}}(PK, S)$.

Convention. Sometimes that we are interested only in a specific part of the output of $Env(n)$ we may use notation such as $(\mathbf{O}, \mathbf{u}, \mathbf{w}, PK) \leftarrow Env(n)$.

The following theorem shows $\mathbf{T}$'s usefulness in breaking seed-circular security.

Theorem 2. *It holds that*

$$\Pr_{\mathbf{Env},\mathbf{T}}[\mathbf{T}(PK, C_1, \ldots, C_n) = S] \geq 1 - \frac{1}{n^5}, \tag{6}$$

where

$$\mathbf{Env} = (\mathbf{O}, \mathbf{u}, \mathbf{w}, S, PK, C_1, \ldots, C_n) \leftarrow Env(n).$$

Proof layout. The proof consists of two parts. First, we show (Lemma 3) that with high probability $S_{out} = S$, where S_{out} is the string decoded in Step 5 of the execution of $\mathbf{T}(PK, C_1, \ldots, C_n)$. Next we show, conditioned on $S_{out} = S$, the probability that $\mathbf{T}(PK, C_1, \ldots, C_n)$ outputs $\perp$ is small (Lemma 4).

Lemma 3. *It holds that*

$$\alpha(n) = \Pr[D^{\widetilde{\mathbf{O}}}(SK', C_1 \ldots C_n) \neq S] \leq \frac{1}{2n^5},$$

where the probability is taken over $(\mathbf{O}, \mathbf{u}, \mathbf{w}, S, PK, C_1, \ldots, C_n) \leftarrow Env(n)$ *and* $(\mathsf{Freq}, SK', \mathsf{Q_s}, \mathsf{Q_c}, \widetilde{\mathbf{O}}) \leftarrow Tvars^{\mathbf{O},\mathbf{u},\mathbf{w}}(PK)$.

Lemma 4. *It holds that*

$$\alpha(n) \stackrel{def}{=} \Pr_{\mathbf{Env},\mathbf{T}}\left[\left(D^{\widetilde{\mathbf{O}}}(SK', C_1 \ldots C_n) = S\right) \wedge (\mathbf{T}(PK, C_1, \ldots, C_n) = \perp)\right] \leq \frac{1}{2^{2n}},$$

where $\mathbf{Env} = (\mathbf{O}, \mathbf{u}, \mathbf{w}, S, PK, C_1, \ldots, C_n) \leftarrow Env(n)$, *and* SK' *and* $\widetilde{\mathbf{O}}$ *are the random variables sampled inside* $\mathbf{T}(PK, C_1, \ldots, C_n)$.

The proof of Theorem 2 follows in a straightforward way by combining Lemmas 3 and 4. We prove Lemma 3 in Subsect. 5.2 and Lemma 4 in Subsect. 5.3.

5.2 Proof of Lemma 3

We start with a simple fact: Informally, it states, in particular, that the string SK' built during an execution of $\mathbf{T}(PK, C_1, \ldots, C_n)$ is a matching secret key of PK relative to $G^{\widetilde{\mathbf{O}}}$.

Fact 3. *For any* $(\mathbf{O}, \mathbf{u}, \mathbf{w}, PK) \in Env(n)$ *and any* $(\mathsf{Freq}, SK', \mathsf{Q_s}, \mathsf{Q_c}, \widetilde{\mathbf{O}}) \in Tvars^{\mathcal{O}}(PK)$*, (a)* $\widetilde{\mathbf{O}}$ *is a correct PKE, and (b)* $(SK', PK) \in G^{\widetilde{\mathbf{O}}}(1^n)$*.*

Equipped with Fact 3, toward proving Lemma 3 we bound the probability that $E^{\mathbf{O}}(PK, S; R) \neq E^{\widetilde{\mathbf{O}}}(PK, S; R)$, for a random R. If this probability is small then with high probability $D^{\widetilde{\mathbf{O}}}(SK', C_1 \ldots C_n)$ results in S, as desired. We will actually bound a related probability, where S above is replaced with 0^n1^n. (Recall that $|S| = n$.) To this end we need the following lemma.

Lemma 5. *Fix* $(\mathbf{O}, \mathbf{u}, \mathbf{w}, PK) \in Env(n)$ *and let* $M = 0^n1^n$*. Let* $(qu_1, \ldots, qu_{2n^{\vartheta+1}})$ *denote the oracle queries asked during the execution of* $E^{\mathbf{O}}(PK, M; R)$*, for a random* R*. Then, for any query index* $1 \leq i \leq 2n^{\vartheta+1}$

$$(A)\ \Pr\left[(qu_i \text{ is } \mathbf{g}\text{- or } \mathbf{e}\text{-type}) \wedge \left(\forall j < i, \mathbf{O}(qu_j) = \widetilde{\mathbf{O}}(qu_j)\right) \wedge \left(\mathbf{O}(qu_i) \neq \widetilde{\mathbf{O}}(qu_i)\right)\right] \leq \frac{1}{n^{8\vartheta}},$$

$$(B)\ \Pr\left[(qu_i \text{ is } \mathbf{d}\text{-type}) \wedge \left(\forall j < i, \mathbf{O}(qu_j) = \widetilde{\mathbf{O}}(qu_j)\right) \wedge \left(\mathbf{d}(qu_i) \neq \widetilde{\mathbf{d}}(qu_i)\right)\right] \leq \frac{1}{n^{8\vartheta}},$$

where $(\mathsf{Freq}, SK', \mathsf{Q_s}, \mathsf{Q_c}, \widetilde{\mathbf{O}}) \leftarrow Tvars^{\mathcal{O}}(PK)$ *and* R *chosen at random.*

We slightly abused notation above by writing $\widetilde{\mathbf{O}}(qu_j)$, since qu_j is a query to $\mathbf{O}$ (e.g., $qu_j = \langle \mathbf{g}, sk' \rangle$); the meaning, however, should be clear.

We first show how to derive Lemma 3 from Lemma 5.

Proof of Lemma 3. All probabilities that appear below are taken over the choices $(\mathbf{O}, \mathbf{u}, \mathbf{w}, S, PK, C_1, \ldots, C_n) \leftarrow Env(n)$ and $(\mathsf{Freq}, SK', \mathsf{Q_s}, \mathsf{Q_c}, \widetilde{\mathbf{O}}) \leftarrow Tvars^{\mathbf{O},\mathbf{u},\mathbf{w}}(PK)$. Let QS be the set of all queries asked during the execution under which $(C_1, \ldots, C_n) \leftarrow E^{\mathbf{O}}(PK, S)$ was produced. We claim

$$\Pr[D^{\widetilde{\mathbf{O}}}(SK', C_1 \ldots C_n) \neq S] \leq \beta(n) \stackrel{\text{def}}{=} \Pr\left[\exists qu \in \mathsf{QS}\colon \mathbf{O}(qu) \neq \widetilde{\mathbf{O}}(qu)\right].$$

The reason is: if the event inside the right-hand side probability does not hold, then $(C_1, \ldots, C_n)$ is also a valid output of $E^{\widetilde{\mathbf{O}}}(PK, S)$. Also, by Fact 3 we know that $(SK', PK) \in G^{\widetilde{\mathbf{O}}}(1^n)$ and that $\widetilde{\mathbf{O}}$ is a correct PKE. Thus, by the correctness of the blackbox construction (G, E, D), we obtain $D^{\widetilde{\mathbf{O}}}(SK', C_1 \ldots C_n) = S$.

Let QS' denote the set of all queries asked during a random execution of $E^{\mathbf{O}}(PK, M)$, where $M = 0^n1^n$. We claim

$$\beta(n) \leq \beta'(n) \stackrel{\text{def}}{=} \Pr[\exists qu \in \mathsf{QS}'\colon \mathbf{O}(qu) \neq \widetilde{\mathbf{O}}(qu)]. \tag{7}$$

Equation 7 holds because: if S has k 0's, then QS is identically distributed to the set of queries asked during a random execution of $E^{\mathbf{O}}(PK, 0^k1^{n-k})$. Moreover, since $k \leq n$, the probability that during a random execution of $E^{\mathbf{O}}(PK, 0^k1^{n-k})$

a query qu, with $\mathbf{O}(qu) \neq \widetilde{\mathbf{O}}(qu)$, is asked is less than the probability that during a random execution of $E^{\mathbf{O}}(PK, M)$ a query qu, with $\mathbf{O}(qu) \neq \widetilde{\mathbf{O}}(qu)$, is asked.

To conclude the proof of Lemma 3 we show $\beta'(n) \leq \frac{1}{2n^5}$. We have

$$\beta'(n) = \Pr\left[\exists qu \in \mathsf{QS}' \colon \mathbf{O}(qu) \neq \widetilde{\mathbf{O}}(qu)\right] \leq 2n^{\vartheta+1} \times \frac{1}{n^{8\vartheta}} \leq \frac{1}{2n^5};$$

the first inequality is obtained by applying Lemma 5 and a union bound. □

We now describe the main lemma and tools we need to prove Lemma 5.

Lemma 6. *For any* $(\mathbf{O}, \mathbf{u}, \mathbf{w}, PK) \in Env(n)$ *and* $(\mathsf{Freq}, SK', \mathsf{Q_s}, \mathsf{Q_c}, \widetilde{\mathbf{O}}) \in Tvars^{\mathcal{O}}(PK)$ *all the following hold: (1) for any* $\mathbf{h} \in \{\mathbf{g}, \mathbf{e}\}$ *if* $(\langle \mathfrak{h}, q\rangle, ans) \in \mathsf{Freq}$*, then* $\mathbf{h}(q) = \widetilde{\mathbf{h}}(q) = ans$*; (2) if* $\mathbf{g}(sk) \neq \widetilde{\mathbf{g}}(sk)$ *for some* sk *then* $(\langle \mathfrak{g}, sk\rangle, *) \in \mathsf{Q_s}$*; (3) if* $\mathbf{e}(pk, b, r) \neq \widetilde{\mathbf{e}}(pk, b, r)$ *for some* pk, b *and* r *then either (a)* $(\langle \mathfrak{e}, (pk, b, r)\rangle, *) \in \mathsf{Q_c}$ *or (b) for some* c*:* $(\langle \mathfrak{e}, (pk, *, *)\rangle, c) \in \mathsf{Q_c}$ *and* $\mathbf{u}(pk, c) = (b, r)$*.*

We require the following standard result [32].

Theorem 4. *(A Chernoff-Hoeffding bound) Let* $x_1, \ldots, x_{n^t}$ *be independent boolean random variables all identically distributed to* x*, and suppose* $\Pr[x = 1] = p$*. Then for* $x_{av} = (x_1 + \cdots + x_{n^t})/n^t$

$$\Pr[|x_{av} - p| \geq \frac{1}{n^k}] \leq \frac{1}{2^{2n^{t-2k}}}. \tag{8}$$

We defer the proof of Lemma 5 to the full version.

5.3 Proof of Lemma 4

Proof. Let $\alpha(n)$ be as in the lemma. To bound $\alpha(n)$, suppose $\mathbf{T}(PK, C_1, \ldots, C_n) = \perp$ and $D^{\widetilde{\mathbf{O}}}(SK', C_1 \ldots C_n) = S$. Then by Step 5 of $\mathbf{T}$'s computation it must hold

$$\mathsf{QPub} \not\subseteq \mathsf{EmbedPub} \cup \mathsf{FreqPub}. \tag{9}$$

Thus,

$$\alpha(n) \leq \Pr_{\mathbf{Env},\mathbf{T}}[\mathsf{QPub} \not\subseteq \mathsf{EmbedPub} \cup \mathsf{FreqPub})], \tag{10}$$

where $\mathbf{Env} = (\mathbf{O}, \mathbf{u}, \mathbf{w}, S, PK, C_1, \ldots, C_n) \leftarrow Env(n)$. We show whenever Eq. 9 holds we can forge a public key in the sense of Lemma 2. Specifically, our forger $\mathcal{B}$, provided with random oracles $(\mathbf{O}, \mathbf{u}, \mathbf{w})$, samples all the variables pertaining to $\mathbf{T}$ by itself and checks whether Eq. 9 holds. Details follow.

The adversary $\mathcal{B}^{\mathbf{O},\mathbf{u},\mathbf{w}}(1^n)$ works as follows:

1. $\mathcal{B}$ samples $S \leftarrow \{0,1\}^n$ and runs $G^{\mathbf{O}}(S)$ to get (SK, PK), and for any query/response $(\langle \mathbf{g}, *\rangle, pk)$ made, it adds pk to $\mathsf{EmbedPub}$.
2. $\mathcal{B}$ samples $\mathsf{Freq} \leftarrow FreqQue^{\mathbf{O},\mathbf{u}}(PK, n^{23\vartheta})$ and then samples $(SK', \mathsf{Q_s}, \mathsf{Q_c})$ by running $ConsOrc(PK, \mathsf{Freq})$.

3. $\mathcal{B}$ forms $\mathsf{FreqPub} = \{pk \mid (\langle \mathbf{g}, * \rangle, pk) \in \mathsf{Freq}\}$ and $\mathsf{QPub} = \{pk \mid (\langle \mathfrak{g}, * \rangle, pk) \in \mathsf{Q_s} \text{ and } \mathbf{w}(pk) = \top\}$. If there is $pk \in \mathsf{QPub} \setminus (\mathsf{EmbedPub} \cup \mathsf{FreqPub})$, $\mathcal{B}$ returns pk; else it returns $pk \leftarrow \{0,1\}^{5n}$.

Let Que be the set of all query/response pairs that $\mathcal{B}$ makes and note that $|\mathsf{Que}|$ is poly-bounded. To analyze $\mathcal{B}$'s success probability, note that for all pk: $pk \in \mathsf{EmbedPub} \cup \mathsf{FreqPub}$ iff $(\langle \mathbf{g}, * \rangle, pk) \in \mathsf{Que}$. Also, by definition if $pk \in \mathsf{QPub}$ then $\mathbf{w}(pk) = \top$. Thus, from Eq. 10, $\mathcal{B}$'s success probability is at least $\alpha(n)$. Applying Lemma 2 the desired bound for $\alpha(n)$ follows. □

5.4 T Does Not Break the CPA Security of the Base Scheme

Theorem 5. *Suppose $\mathcal{A}$ is a CPA adversary with access to oracles $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d})$ and $\mathbf{T}$ that makes at most $2^{n/8}$ queries. We have*

$$\Pr_{\mathcal{O},\mathbf{T},b,sk,c}[\mathcal{A}^{\mathbf{g},\mathbf{e},\mathbf{d},\mathbf{T}}(1^n, pk, c) = b] \leq \frac{1}{2} + \frac{1}{2^{n/4}}, \tag{11}$$

where $\mathcal{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{u}, \mathbf{w}) \leftarrow \Psi$, $b \leftarrow \{0,1\}$, $sk \leftarrow \{0,1\}^n$, $pk = \mathbf{g}(sk)$ and $c \leftarrow \mathbf{e}(pk, b)$.

The following lemma is used in the proof of Theorem 5: it shows how to simulate responses to queries to $\widetilde{\mathbf{O}}$ via oracle access to $(\mathbf{O}, \mathbf{u}, \mathbf{w})$.

Lemma 7. *Fix $(\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{u}, \mathbf{w}, PK) \in Env(n)$ and $(\mathsf{Freq}, SK', \mathsf{Q_s}, \mathsf{Q_c}, \widetilde{\mathbf{g}}, \widetilde{\mathbf{e}}, \widetilde{\mathbf{d}}) \in Tvars^{\mathcal{O}}(PK)$. Assuming $\mathsf{Q_c} = \{(\langle \mathfrak{e}, (pk_1, b_1, r_1) \rangle, c_1), \ldots, (\langle \mathfrak{e}, (pk_p, b_p, r_p) \rangle, c_p)\}$ let $\mathsf{W} = \{(pk_i, c_i) \colon 1 \leq i \leq p\}$ and $\mathsf{W}' = \{(pk_i, \mathbf{e}(pk_i, b_i, r_i)) \colon 1 \leq i \leq p\}$. Then*

(a) Both $\widetilde{\mathbf{g}}$ and $\widetilde{\mathbf{e}}$ can be computed efficiently (on all points) given access to oracles $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d})$ and having $\mathsf{Q_s}$ and $\mathsf{Q_c}$ as input.

(b) For any (sk, c), if $(\langle \mathfrak{g}, sk \rangle, *) \notin \mathsf{Q_s}$, the value $\widetilde{\mathbf{d}}(sk, c)$ can be efficiently computed given access to oracle $\mathbf{O}$ and having $\mathsf{Q_c}$ as input.

(c) If for (sk, c) it holds that $(\langle \mathfrak{g}, sk \rangle, pk) \in \mathsf{Q_s}$ for some pk and that $(pk, c) \notin \mathsf{W} \cup \mathsf{W}'$, then $\widetilde{\mathbf{d}}(sk, c)$ can be determined as follows: if $\mathbf{u}(pk, c) = (b, *) \neq \perp$ then $\widetilde{\mathbf{d}}(sk, c) = b$; otherwise, $\widetilde{\mathbf{d}}(sk, c) = \perp$.

(d) If for (sk, c) it holds that $(\langle \mathfrak{g}, sk \rangle, pk) \in \mathsf{Q_s}$, for some pk, and that $(pk, c) \in \mathsf{W}' \setminus \mathsf{W}$, then $\widetilde{\mathbf{d}}(sk, c) = \perp$.

(e) If for (sk, c) it holds that $(\langle \mathfrak{g}, sk \rangle, pk) \in \mathsf{Q_s}$, for some pk, and that $(pk, c) = (pk_i, c_i)$ for some $i \leq p$, then $\widetilde{\mathbf{d}}(sk, c) = b_i$.

Proof sketch of Theorem 5. As in Sect. 4 the idea is to give an adversary $\mathcal{B}$, where $\mathcal{B}^{\mathcal{O}}(1^n, pk, c)$ can simulate responses to $\mathbf{T}$ queries of $\mathcal{A}^{\mathbf{O},\mathbf{T}}(1^n, pk, c)$, without calling $\langle \mathbf{u}, (pk, c) \rangle$. Let $Tqu = \langle \mathbf{T}, (1^n, PK, C_1, \ldots, C_n) \rangle$ be an $\mathcal{A}$'s query. As per $\mathbf{T}$'s computation, $\mathcal{B}$ first samples $\mathsf{Freq} \leftarrow FreqQue^{\mathbf{O},\mathbf{u}}(PK, n^{23\vartheta})$. This may seem problematic since this step involves making $\mathbf{u}$ queries. By inspecting Definition 6, however, we can see for any query $\langle \mathbf{u}, (pk', *) \rangle$ that needs to be made, $\mathcal{B}$ already knows $\mathbf{g}^{-1}(pk')$. Finally, let $\mathsf{FreqPub}$ contain any pk' such that

$(\langle \mathfrak{g}, * \rangle, pk') \in \mathsf{Freq}$, and assume w.l.o.g. $pk \notin \mathsf{FreqPub}$ (because otherwise $\mathcal{B}$ has already found $\mathbf{g}^{-1}(pk)$.)

Next, $\mathcal{B}$ samples $(SK', \mathsf{Q_s}, \mathsf{Q_c}, \widetilde{\mathbf{O}})$ as in $\mathbf{T}$'s execution and starts simulating $D^{\widetilde{\mathbf{O}}}(SK', C_1 \ldots, C_n)$. Again the idea is to see if $(\langle \mathfrak{g}, * \rangle, pk) \in \mathsf{Q_s}$ or not. If $(\langle \mathfrak{g}, * \rangle, pk) \notin \mathsf{Q_s}$: by Lemma 7 we can see $\mathcal{B}$ can handle all $\widetilde{\mathbf{O}}$ queries. In particular, $\mathcal{B}$ will never need to call $\langle \mathbf{u}, (pk, *) \rangle$. After the decryption $\mathcal{B}$ performs Step 5 of $\mathbf{T}$'s computation, which $\mathcal{B}$ can efficiently do, since no $\mathbf{u}$ queries are involved.

If, however, $(\langle \mathfrak{g}, * \rangle, pk) \in \mathsf{Q_s}$, assuming $S_{out} = D^{\widetilde{\mathbf{O}}}(SK', C_1 \ldots, C_n)$, since $pk \notin \mathsf{FreqPub}$, the answer to Tqu is $\perp$ unless $(\langle \mathbf{g}, * \rangle, pk)$ occurs during $G^{\mathbf{O}}(S_{out})$. Now as before the idea is to show $\mathcal{B}$ can find S_0 and S_1 s.t. $S_{out} \in \{S_0, S_1\}$. To this end $\mathcal{B}$ starts simulating $D^{\widetilde{\mathbf{O}}}(SK', C_1 \ldots, C_n)$. By Lemma 7 all $\widetilde{\mathbf{g}}$ and $\widetilde{\mathbf{e}}$ queries can be handled. Let the sets W and W' be formed based on $\mathsf{Q_c}$ as in Lemma 7. For $\widetilde{\mathbf{d}}$ queries: $\mathcal{B}$ will be unable to simulate the answer to a query $qu = \widetilde{\mathbf{d}}(sk', c')$ only if $(\langle \mathfrak{g}, sk' \rangle, pk) \in \mathsf{Q_s}$, $c' = c$ and $(pk, c) \notin \mathsf{W} \cup \mathsf{W}'$ (Case (c) of Lemma 7): in this case, knowing that the answer is the challenge bit b, $\mathcal{B}$ starts two branches of simulation, where it replies to qu with 0 on one branch and with 1 on the other. As before, we need to make sure $\mathcal{B}$ provides a consistent reply on either branch if the same query shows up in the future. The two strings decoded on the two branches at the end satisfy the above claim. □

5.5 Putting All Together

We may now use our two main established results to obtain our main result.

Theorem 6. *There exists no fully-blackbox construction of* 1*-seed-circular-secure bit-encryption schemes from CPA-secure encryption schemes.*

5.6 Extensions of the Separation Result

We briefly discuss why our separation holds even if E is a $(c \log n)$-bit encryption scheme and where our separation fails if E is allowed to be full length.

Remark 1. We first sketch how to adjust $\mathbf{T}$ to make our separation work for the case that (G, E, D) is an η-bit PKE, for $\eta = c \log n$. To this end, we need to change Definition 6 (i.e., the procedure $FreqQue$), so that instead of encrypting 0 and 1 many times (as in the bit encryption case), it encrypts all messages $m \in \{0,1\}^\eta$, each many times. The total number of queries still remains polynomial. The description of the oracle $\mathbf{T}$ remains unchanged except that in the first step we call this new version of $FreqQue$. We can now prove the exact same version as Lemma 5, by changing M to have n copies of each string $z \in \{0,1\}^\eta$ (instead of having n copies of 0 and n of 1 as in the bit-encryption case). Now the proofs of the rest of the lemmas that lead up to Theorem 2, as well as the proof of Theorem 5, remain unchanged.

Finally, note that the above extension heavily relies on the fact that the message size is $O(\log n)$, and so it does not apply if the constructed scheme is full-length, as expected.

6 Bit t-seed-circular Security $\not\Rightarrow$ Full-Length $(t+1)$-seed-circular Security

In this section we present our results for separating full-length $(t+1)$-seed-circular security from bit t-seed-circular security. To this end we define a weakening oracle $\mathbf{T_{t+1}}$, for a fixed candidate construction (G, E, D), generalizing a similar oracle given in Subsect. 4.2. Throughout this section note that (G, E, D) has the same plaintext and seed space. Mots of the tools underlying $\mathbf{T_{t+1}}$ have been presented before, but we need the following extension of Definition 6.

Definition 9. *We define the following probabilistic procedure, $ExtFreqQue$.*

- ***Oracles:*** $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{u})$
- ***Input:*** *A security parameter 1^n (left implicit), public key PK and $p \in \mathbb{N}$.*
- ***Output:*** *A set* Freq *formed as follows.*
  - *Do the following independently p times and add the symbolic versions of all query/response pairs to* Freq*: Sample $S \leftarrow \{0,1\}^n$, and run $G^{\mathbf{O}}(S)$ and $E^{\mathbf{O}}(1^n, PK, S)$.*
  - *Finally, for any $(\langle \mathfrak{d}, (sk, c)\rangle, *) \in \mathsf{Freq}$ if $\mathbf{u}(\mathbf{g}(sk), c) = (b, r) \neq \perp$ add $(\langle \mathfrak{e}, (pk, b, r)\rangle, c)$ to* Freq.

Remark 2. Throughout the remaining sections we continue to use Assumption 1. In particular, since our focus right now is on schemes (G, E, D) with plaintext space $\{0,1\}^n$ (i.e., the same as the seed space) we assume that E on any plaintext $m \in \{0,1\}^n$ makes exactly n^{ϑ} queries.

Description of $\mathbf{T}_{t+1}$: We present the oracle $\mathbf{T}_{t+1}$. This new oracle shares many aspects with the oracle $\mathbf{T}$, and so we leave out details whenever appropriate.
Notation. Let t_0 be such that $t \leq n^{t_0}$.
Oracles: $\mathcal{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{u}, \mathbf{w})$. Denote $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d})$.
Input: $(1^n, PK_1, \ldots, PK_{t+1}, C_1, \ldots, C_{t+1})$

1. **Learning heavy queries:** For $i \leq t+1$ let $\mathsf{Freq_i} \leftarrow ExtFreqQue^{\mathbf{O},\mathbf{u}}(PK_i, n^{23\vartheta+4t_0})$, and let $\mathsf{FreqPub_i}$ be the set of public keys pk s.t. $(\langle \mathfrak{g}, *\rangle, pk) \in \mathsf{Freq_i}$.
2. **Sampling consistent oracles/secret-keys:** For $i \leq t+1$ sample
$$(\widetilde{SK}_i, \mathsf{Q}^{\mathsf{i}}_{\mathsf{s}}, \mathsf{Q}^{\mathsf{i}}_{\mathsf{c}}) \leftarrow ConsOrc(PK_i, \mathsf{Freq_i}), \tag{12}$$
and let $\mathsf{QPub_i}$ contain any pk such that $\mathbf{w}(pk) = \top$ and $(\langle \mathfrak{g}, *\rangle, pk) \in \mathsf{Q}^{\mathsf{i}}_{\mathsf{s}}$.
3. If for some distinct $i, j \in [t+1]$ $(\mathsf{QPub_i} \cap \mathsf{QPub_j}) \setminus (\mathsf{FreqPub_i} \cup \mathsf{FreqPub_j}) \neq \emptyset$ halt and return $\perp$.
4. **Defining intermediate oracles:** For $i \leq t+1$ define
$$(\mathbf{e}_{imp,i}, \mathbf{d}_{imp,i}) = EncImpose(\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathsf{Q}^{\mathsf{i}}_{\mathsf{c}})$$
$$(\widetilde{\mathbf{g}}_i, \widetilde{\mathbf{d}}_i) = KeyImpose(\mathbf{g}, \mathbf{e}_{imp,i}, \mathbf{d}_{imp,i}, \mathsf{Q}^{\mathsf{i}}_{\mathsf{s}}).$$
Let $\widetilde{\mathbf{e}}_i = \mathbf{e}_{imp,i}$, and $\widetilde{\mathbf{O}}_i = (\widetilde{\mathbf{g}}_i, \widetilde{\mathbf{e}}_i, \widetilde{\mathbf{d}}_i)$.

5. **Decrypting the ciphertexts:** Set $S^1_{out} = D^{\widetilde{\mathbf{O}_{t+1}}}(\widetilde{SK_{t+1}}, C_{t+1})$ and for $2 \leq i \leq t+1$ set $S^i_{out} = D^{\widetilde{\mathbf{O}_{i-1}}}(\widetilde{SK_{i-1}}, C_{i-1})$.
6. **Forming the output.** For $i \leq t+1$ run $G^{\mathbf{O}}(S^i_{out})$ and let $\mathsf{EmbedPub_i}$ contain any pk such that the query/response $(\langle \mathbf{g}, * \rangle, pk)$ is made during the execution. If for all $i \leq t+1$, $\mathsf{QPub_i} \subseteq \mathsf{EmbedPub_i} \cup \mathsf{FreqPub_i}$, then return S^1_{out}; otherwise, return $\perp$.

To state the main results we define the following environment, specifying a random choice of $(\mathbf{O}, \mathbf{u}, \mathbf{w})$ plus those underlying an honest input to $\mathbf{T_{t+1}}$.

Environment: $Env_t(n)$: Output

$$(\mathbf{O}, \mathbf{u}, \mathbf{w}, S_1, \ldots, S_t, PK_1, \ldots, PK_t, E^{\mathbf{O}}(PK_1, S_2), \ldots, E^{\mathbf{O}}(PK_t, S_1)),$$

where $(\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{u}, \mathbf{w}) \leftarrow \Psi$, $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d})$, $S_i \leftarrow \{0,1\}^n$ and $(SK_i, PK_i) = G^{\mathbf{O}}(S_i)$, for $1 \leq i \leq t$.

We now state the two main results leading to our claimed separation.

Theorem 7. *It holds that*

$$\Pr_{\mathbf{Env}, \mathbf{T_{t+1}}}[\mathbf{T_{t+1}}(PK_1, \ldots, PK_{t+1}, C_1, \ldots, C_{t+1}) = S_1] \geq 1 - \frac{1}{n^5}, \tag{13}$$

where

$$\mathbf{Env} = (\mathbf{O}, \mathbf{u}, \mathbf{w}, S_1, \ldots, S_{t+1}, PK_1, \ldots, PK_{t+1}, C_1, \ldots, C_{t+1}) \leftarrow Env_{t+1}(n).$$

Theorem 8. *Suppose $\mathcal{A}$ is a t-seed circular security adversary with access to oracles $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d})$ and $\mathbf{T}$ that makes at most $2^{n/8}$ queries. We have*

$$\Pr_{\mathcal{O}, \mathbf{T}, b, sk_1, \ldots, sk_t}[\mathcal{A}^{\mathbf{O}, \mathbf{T}}(1^n, pk_1, \ldots, pk_t, \mathbf{e}(pk_1, sk_2), \ldots, \mathbf{e}(pk_t, sk_1), \mathbf{e}(pk_1, b)) = b] \leq \frac{1}{2} + \frac{1}{2^{n/4}},$$

where $\mathcal{O} = (\mathbf{O}, \mathbf{u}, \mathbf{w}) \leftarrow \Psi$, $b \leftarrow \{0,1\}$, $sk_1, \ldots, sk_t \leftarrow \{0,1\}^n$ and $pk_i = \mathbf{g}(sk_i)$ for $i \leq t$.

By combining the above two theorems we have the following.

Theorem 9. *There exists no fully-blackbox construction of full-length $(t+1)$-seed-circular secure encryption from t-seed-circular secure bit encryption.*

7 Constructions Based on Circular Security

We show how our results on seed-circular security extend to rule out a class of constructions for circular security that we call *key-isolating constructions*. To define this class we first define it in a related model we call the *canonical model*, and then we define it in the standard model. We start with some definitions.

Canonical-Form (CF) PKE. We call $\mathbf{O} = (\mathbf{gs}, \mathbf{gp}, \mathbf{e}, \mathbf{d})$ a CF PKE if the domain of $\mathbf{gp}$ (excluding 1^n) is the range of $\mathbf{gs}$ and $(\mathbf{g}, \mathbf{e}, \mathbf{d})$, where

$\mathbf{g}(s) = (\mathbf{gs}(s), \mathbf{gp}(\mathbf{gs}(s)))$, is a PKE. That is, the key-generation algorithm of a CF scheme first deterministically maps a seed to a secret key, and then *deterministically* maps the secret key to a public key.

CF-based blackbox model. A blackbox construction in the CF model is a tuple of oracle algorithms (GS, GP, E, D) s.t. for any CF PKE $\mathbf{O} = (\mathbf{gs}, \mathbf{gp}, \mathbf{e}, \mathbf{d})$, $(GS^{\mathbf{O}}, GP^{\mathbf{O}}, E^{\mathbf{O}}, D^{\mathbf{O}})$ is a CF PKE. Proving a syntactically-unrestricted impossibility result in the CF model implies one in the standard model, since any CPA-secure CF PKE can be turned into a CPA-secure standard PKE and that any circular-secure standard PKE can be put into a circular-secure CF PKE.

CF Key-isolating constructions. We call (GS, GP, E, D) *key-isolating* if GS never calls $\mathbf{gp}$ of the base scheme, i.e., GS only has access to $(\mathbf{gs}, \mathbf{e}, \mathbf{d})$.

Ruling-out key-isolating constructions. Our earlier results extend to rule out CF key-isolating constructions for circular security. To do this, we first need to change the distribution of Ψ, by replacing $\mathbf{g}$ with $(\mathbf{gs}, \mathbf{gp})$, for $\mathbf{gs}_n \colon \{0,1\}^n \to \{0,1\}^{3n}$ and $\mathbf{gp}_n \colon \{0,1\}^{3n} \to \{0,1\}^{5n}$. As for $\mathbf{T}$, which now takes as input a public key and an encryption of a PK's secret key, all we need to change is that in Step 5 of $\mathbf{T}$'s description the set $\mathsf{EmbedPub}$ should be formed by executing $GP^{\mathbf{O}}$ on the intermediate, decrypted string (which is now a secret key). All our proofs about $\mathbf{T}$ not breaking the semantic security of the base scheme go through with only making obvious modifications. The proofs about $\mathbf{T}$ being helpful in breaking the circular security of the constructed scheme follow by noting that all access to $\mathbf{gp}$ during key generation is only made by GP. This fact only becomes essential in the proof of Lemma 4, and is the reason behind the above way of defining $\mathsf{EmbedPub}$. Other lemmas follow by making only obvious changes.

Interpretation w.r.t. standard constructions. Our above result also rules out *standard key-isolating* constructions. To define this notion for a standard construction (G, E, D) we first need to slightly change the standard model so that (G, E, D) takes as oracles a CF PKE. Again, as explained above this is w.l.o.g. Now we call $\mathcal{E} = (G, E, D)$ a *standard key-isolating construction* if $\mathcal{E}$ admits a key-isolating CF *counterpart* in the following sense: there exists algorithms GS and GP s.t. (GS, GP, E, D) is key-isolating and (GS, GP) induces the same distribution as G, i.e., for any $\mathbf{O} = (\mathbf{gs}, \mathbf{gp}, \mathbf{e}, \mathbf{d})$ it holds that (SK, PK) is identically distributed to (SK', PK'), where $(SK, PK) \leftarrow G^{\mathbf{O}}(1^n)$, $SK' \leftarrow GS^{\mathbf{gs},\mathbf{e},\mathbf{d}}(1^n)$ and $PK' = GP^{\mathbf{O}}(SK')$. Now the impossibility of CF key-isolating constructions extends to standard ones, by how the notion counterpart is defined.

Examples. Any standard construction $\mathcal{E} = (G, E, D)$ under which seeds and secret keys are the same is key-isolating: defining $GS(S) = S$ and $GP^{\mathbf{gs},\mathbf{gp},\mathbf{e},\mathbf{d}}(S) = G_2^{\mathbf{gs},\mathbf{gp},\mathbf{e},\mathbf{d}}(S)$, where G_2 is the algorithm corresponding to the public-key output of G, the construction (GS, GP, E, D) is the CF-counterpart of (G, E, D) and is key-isolating since GS makes no oracle calls at all. The class of key-isolating constructions is larger than this; we only wanted to give a concrete example.

We leave a more comprehensive and formal discussion to the full version.

8 Discussion Related to Impossibility for Circular Security

In this section we briefly explain why we were not able to fully extend our results to the circular-security case. For simplicity, we highlight the difficulties encountered w.r.t. the simple type of constructions discussed in Sect. 4. In what follows all mentions of the oracle $\mathbf{T}$ for the seed-circular-security case refers to the oracle $\mathbf{T}$ defined in Sect. 4.

As discussed previously, the main challenge in designing an appropriate Oracle $\mathbf{T}$ is to make sure that responses to queries to $\mathbf{T}$ do not leak information about the challenge secrets of a CPA adversary $\mathcal{A}$ against $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d})$. We proved this for the seed-circular-security case by providing a CCA2 adversary $\mathcal{B}$ in such a way that $\mathcal{B}^{\mathbf{O},\mathbf{u},\mathbf{w}}(pk, c)$ is able to simulate $\mathcal{A}^{\mathbf{O},\mathbf{T}}(pk, c)$.

Roughly speaking, the only part of the execution of $\mathbf{T}(PK, C_1, \ldots, C_n)$ that is not simulatable by a CCA2-adversary $\mathcal{B}^{\mathbf{O},\mathbf{u},\mathbf{w}}(pk, c)$ is when during the computation of $D^{\widetilde{\mathbf{d}}}(SK', C_1 \ldots C_n)$ a query $\widetilde{\mathbf{d}}(sk', c)$ shows up and $(\langle \mathfrak{g}, sk' \rangle, pk) \in \mathsf{Q_s}$. We fixed this non-simulatability problem by adding an extra check at the end of $\mathbf{T}$'s computation that ensures the following: either the value of $\mathbf{g}^{-1}(pk)$ is *embedded* in S_{out} (i.e., the query/response pair $(\langle \mathbf{g}, * \rangle, pk)$ shows up during $G^{\mathbf{g}}(S_{out})$) or the answer to the underlying $\mathbf{T}$ query is $\perp$.

To define a circular-security weakening oracle $\mathbf{T}$ we may be tempted to proceed as before: $\mathbf{T}$ accepts inputs of the form $(PK, C_1, \ldots, C_n)$, where now $C_1, \ldots, C_n$ are (supposedly) bit-wise encryption of a PK's secret key under PK itself. (For simplicity, assume that the length of the secret key is n.) Then, everything remain unchanged, as in $\mathbf{T}$ in Sect. 4, until $\mathbf{T}$ obtains $S_{out} = D^{\widetilde{\mathbf{d}}}(SK', C_1 \ldots C_n)$, which now is supposedly a PK's matching secret key. Now in order to make sure that the oracle $\mathbf{T}$ is simulatable (i.e., it does not leak non-simulatable information to a CPA adversary against $\mathbf{O}$) it seems that, as before, we need to make sure that $\mathbf{g}^{-1}(\mathsf{Q_{pub}})$ is "embedded" in S_{out}, before releasing S_{out}. (Recall the definition of $\mathsf{Q_{pub}}$ from $\mathbf{T}$'s definition in Sect. 4.) But this "embedding condition" seems hard to check. This check was easy for the seed-circular-security case since we can simply run $G^{\mathbf{g}}(S_{out})$ and monitor all sk' for which we observe a query/response $(\langle \mathbf{g}, sk' \rangle, *)$. For the circular-security case one idea is to run $D^{\mathbf{d}}(S_{out}, \cdot)$ on many random encryptions produced as $C \leftarrow E^{\mathbf{e}}(PK, b; R)$ for randomly chosen b and R, and record in a set $\mathsf{EmbedSec}$ all sk' for which we encounter a query $\langle \mathbf{d}, (sk', *) \rangle$. We then return S_{out} if $\mathsf{Q_{pub}} \subseteq \mathbf{g}(\mathsf{EmbedSec})$; otherwise, we return $\perp$. While this check makes the oracle $\mathbf{T}$ simulatable, it makes $\mathbf{T}$ unfortunately too weak in that we cannot anymore guarantee in general that $\mathbf{T}(PK, C_1 \ldots C_n)$ will return SK with non-negligible probability, for $(SK, PK) \leftarrow G^{\mathbf{g}}(1^n)$ and $(C_1, \ldots, C_n) \leftarrow E^{\mathbf{e}}(PK, SK)$, i.e., $\mathbf{T}$ is not useful in general for breaking circular security. (Contrived constructions (G, E, D) for which this is the case can be given.)

Acknowledgements. We would like to thank Mohammad Mahmoody for useful conversations in an early stage of this work.

References

1. Alamati, N., Peikert, C.: Three's compromised too: circular insecurity for any cycle length from (Ring-)LWE. In: Robshaw and Katz [37], pp. 659–680
2. Applebaum, B.: Key-dependent message security: generic amplification and completeness. J. Cryptol. **27**(3), 429–451 (2014)
3. Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 595–618. Springer, Heidelberg (2009). doi:10.1007/978-3-642-03356-8_35
4. Asharov, G., Segev, G.: Limits on the power of indistinguishability obfuscation and functional encryption. In: Guruswami, V. (ed.) FOCS 2015, pp. 191–209. IEEE Computer Society (2015)
5. Baecher, P., Brzuska, C., Fischlin, M.: Notions of black-box reductions, revisited. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013. LNCS, vol. 8269, pp. 296–315. Springer, Heidelberg (2013). doi:10.1007/978-3-642-42033-7_16
6. Bishop, A., Hohenberger, S., Waters, B.: New circular security counterexamples from decision linear and learning with errors. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9453, pp. 776–800. Springer, Heidelberg (2015). doi:10.1007/978-3-662-48800-3_32
7. Boneh, D., Halevi, S., Hamburg, M., Ostrovsky, R.: Circular-secure encryption from decision Diffie-Hellman. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 108–125. Springer, Heidelberg (2008). doi:10.1007/978-3-540-85174-5_7
8. Boneh, D., Papakonstantinou, P.A., Rackoff, C., Vahlis, Y., Waters, B.: On the impossibility of basing identity based encryption on trapdoor permutations. In: FOCS 2008, pp. 283–292. IEEE Computer Society (2008)
9. Brakerski, Z., Goldwasser, S.: Circular and leakage resilient public-key encryption under subgroup indistinguishability. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 1–20. Springer, Heidelberg (2010). doi:10.1007/978-3-642-14623-7_1
10. Brakerski, Z., Katz, J., Segev, G., Yerukhimovich, A.: Limits on the power of zero-knowledge proofs in cryptographic constructions. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 559–578. Springer, Heidelberg (2011). doi:10.1007/978-3-642-19571-6_34
11. Canetti, R., Lin, H., Tessaro, S., Vaikuntanathan, V.: Obfuscation of probabilistic circuits and applications. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9015, pp. 468–497. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46497-7_19
12. Cash, D., Green, M., Hohenberger, S.: New definitions and separations for circular security. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 540–557. Springer, Heidelberg (2012). doi:10.1007/978-3-642-30057-8_32
13. Cheng, C.-M., Chung, K.-M., Persiano, G., Yang, B.-Y. (eds.): PKC 2016, (II). LNCS, vol. 9615. Springer, Heidelberg (2016)
14. Choi, S.G., Dachman-Soled, D., Malkin, T., Wee, H.: Black-box construction of a non-malleable encryption scheme from any semantically secure one. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 427–444. Springer, Heidelberg (2008). doi:10.1007/978-3-540-78524-8_24
15. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography (extended abstract). In: STOC 1991, pp. 542–552 (1991)
16. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Mitzenmacher, M. (ed.) STOC 2009, pp. 169–178. ACM (2009)

17. Gertner, Y., Kannan, S., Malkin, T., Reingold, O., Viswanathan, M.: The relationship between public key encryption and oblivious transfer. In: FOCS 2000, pp. 325–335. IEEE Computer Society (2000)
18. Gertner, Y., Malkin, T., Myers, S.: Towards a separation of semantic and CCA security for public key encryption. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 434–455. Springer, Heidelberg (2007). doi:10.1007/978-3-540-70936-7_24
19. Gertner, Y., Malkin, T., Reingold, O.: On the impossibility of basing trapdoor functions on trapdoor predicates. In: FOCS 2001, pp. 126–135. IEEE Computer Society (2001)
20. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game. In: Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing, pp. 218–229. ACM (1987)
21. Haitner, I., Holenstein, T.: On the (Im)Possibility of key dependent encryption. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 202–219. Springer, Heidelberg (2009). doi:10.1007/978-3-642-00457-5_13
22. Hajiabadi, M., Kapron, B.M.: Reproducible circularly-secure bit encryption: applications and realizations. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 224–243. Springer, Heidelberg (2015). doi:10.1007/978-3-662-47989-6_11
23. Hajiabadi, M., Kapron, B.M., Srinivasan, V.: On generic constructions of circularly-secure, leakage-resilient public-key encryption schemes. In: Cheng et al.[13], pp. 129–158
24. Hsiao, C.-Y., Reyzin, L.: Finding collisions on a public road, or do secure hash functions need secret coins? In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 92–105. Springer, Heidelberg (2004). doi:10.1007/978-3-540-28628-8_6
25. Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: Johnson, D.S. (ed.) STOC 1989, pp. 44–61. ACM (1989)
26. Ishai, Y., Kushilevitz, E., Lindell, Y., Petrank, E.: Black-box constructions for secure computation. In: Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing, pp. 99–108. ACM (2006)
27. Koppula, V., Waters, B.: Circular security separations for arbitrary length cycles from LWE. In: Robshaw and Katz [37], pp. 681–700
28. Mahmoody, M., Mohammed, A.: On the power of hierarchical identity-based encryption. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9666, pp. 243–272. Springer, Heidelberg (2016). doi:10.1007/978-3-662-49896-5_9
29. Mahmoody, M., Pass, R.: The curious case of non-interactive commitments – on the power of black-box vs. non-black-box use of primitives. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 701–718. Springer, Heidelberg (2012). doi:10.1007/978-3-642-32009-5_41
30. Malkin, T., Teranishi, I., Yung, M.: Efficient circuit-size independent public key encryption with KDM security. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 507–526. Springer, Heidelberg (2011). doi:10.1007/978-3-642-20465-4_28
31. Marcedone, A., Pass, R., Shelat, A.: Bounded KDM security from iO and OWF. In: Zikas, V., Prisco, R. (eds.) SCN 2016. LNCS, vol. 9841, pp. 571–586. Springer, Cham (2016). doi:10.1007/978-3-319-44618-9_30
32. Motwani, R., Raghavan, P.: Randomized Algorithms. Cambridge University Press, New York (1995)
33. Myers, S., Shelat, A.: Bit encryption is complete. In: Foundations of Computer Science, 2009, FOCS 2009, pp. 607–616. IEEE (2009)

34. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing, pp. 427–437. ACM (1990)
35. Pass, R., Shelat, A., Vaikuntanathan, V.: Construction of a non-malleable encryption scheme from any semantically secure one. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 271–289. Springer, Heidelberg (2006). doi:10.1007/11818175_16
36. Reingold, O., Trevisan, L., Vadhan, S.: Notions of reducibility between cryptographic primitives. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 1–20. Springer, Heidelberg (2004). doi:10.1007/978-3-540-24638-1_1
37. Robshaw, M., Katz, J. (eds.): CRYPTO 2016 (II). LNCS, vol. 9815. Springer, Heidelberg (2016)
38. Rosen, A., Segev, G.: Chosen-ciphertext security via correlated products. SIAM J. Comput. **39**(7), 3058–3088 (2010)
39. Rothblum, R.D.: On the circular security of bit-encryption. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 579–598. Springer, Heidelberg (2013). doi:10.1007/978-3-642-36594-2_32
40. Simon, D.R.: Finding collisions on a one-way street: can secure hash functions be based on general assumptions? In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 334–345. Springer, Heidelberg (1998). doi:10.1007/BFb0054137
41. Vahlis, Y.: Two is a crowd? a black-box separation of one-wayness and security under correlated inputs. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 165–182. Springer, Heidelberg (2010). doi:10.1007/978-3-642-11799-2_11
42. Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 24–43. Springer, Heidelberg (2010). doi:10.1007/978-3-642-13190-5_2
43. Wee, H.: KDM-security via homomorphic smooth projective hashing. In: Cheng et al. [13], pp. 159–179