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

Mohammad Hajiabadi and Bruce M. Kapron

[1] Department of Computer Science, University College 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, for $t \geq 2$, all constructions of t-circular-secure 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* public-key bit encryption from CPA-secure public-key encryption;
- t-seed circular secure public-key encryption from $(t-1)$-seed circular secure public-key encryption, for any $t \geq 2$.

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 separation model follows that of Gertner, Malkin and Reingold (FOCS'01), which is a weaker separation model than that of Impagliazzo and Rudich.

* A preliminary version of this work will appear in the proceedings of Eurocrypt 2017. Work supported in part by an NSERC Discovery Grant. Part of this work was completed while the first author was at University College London and received funding from the European Research Council under the ERC Grant Agreement n. 307937.

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 $pk_{(i+1 \mod t)}$. These notions are a special case of the notion of key-dependent-message (KDM) security, under which more general functions of the secret key(s) may be encrypted.

A primary foundational application of the notion of circular security (for any t) is in the context of fully homomorphic encryption (FHE). Currently, with the exception of [CLTV15], 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 (e.g., [Gen09,VDGHV10,BV11,BV14,GSW13,BGV14]).

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.[3] 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 [BHHO08,ACPS09,BG10,MTY11,Wee16,MPS16]. The main challenge involved is that the secret key of a public key should now "communicate" with another public key in a way that a key cycle reveals no information about the underlying secret keys. Implementing such an idea based on structure-free general primitives has been unfruitful. One of the goals of our work is to formally 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 by design is encrypted bit-by-bit or block-by-block, where the size of each block is considerably smaller than the secret-key size (e.g., [Gen09,VDGHV10]). In such cases the above folklore construction (for $t = 1$) fails: 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 [App14], *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.

A series of papers, some quite recent, based on various specific assumptions give schemes that are CPA secure, but not t-circular secure (for various values of t) [CGH12,BHW15,KW16,AP16]. While these results provide strong 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 work addressing the problem. Haitner and Holenstein [HH09] rule out fully-blackbox constructions of KDM-secure encryption w.r.t. quite large function families from trapdoor permutations. Rothblum [Rot13] 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 [Rot13] only considers reductions to and from the same scheme, as opposed to our results which deal with constructions.

Before moving on, we remind the reader of the simple fact that bit t-circular security implies full-length t-circular security (see Terminology 1). Briefly, the state of knowledge regarding circular security can be summarized as follows:

[3] Assume, w.l.o.g, the base 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.

– Full-length t-circular security based on CPA security: we have a simple construction for $t = 1$, but no known constructions for $t > 1$.
– Bit t-circular security: all constructions (for any t) are based on specific assumptions [BHHO08,BG10] and there is a preliminary blackbox separation for $t = 1$ from CPA security [Rot13].

In this work we ask the following two questions
(I) Can bit 1-circular security be based on CPA security?
(II) 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 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; see Terminology 1. 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 and changing the decryption algorithm accordingly. 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 [RTV04]) of 1-*seed circular secure* public-key bit encryption from CPA-secure public-key encryption (Theorem 10). 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 (Section 5.8).
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 13).

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 1-circular-secure bit encryption that we call *key-isolating constructions* (Section 7). For example, the impossibility result of [Rot13] is obtained as a special case of our septation for the circular-security setting. Moreover, we show that our separation extends even if the base encryption scheme is CCA2 secure and the constructed scheme is only required to satisfy a weak form of 1-seed circular security, under which the adversary should recover the seed (Section 5.8).

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 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 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 even a less ambitious goal than that of Question (II), 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 reduction is, however, non-blackbox.

From a practical point of view, the notion of seed circular security for specific schemes is not natural since such schemes typically come with *public parameters* (e.g., a group), and it is not 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. For example, both circular-secure schemes of [BHHO08,BG10] 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 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 [GMR01]. We discuss the model for the first result. For any candidate 1-seed circular-secure bit-encryption 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 1-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 Section 4.

Most separation results in the literature indeed rule out the existence of *relativizing reductions*, e.g., [IR89,Sim98,GKM$^+$00,BPR$^+$08,Vah10], 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., [HR04,HH09,MP12,MM16].

Comparison with [HH09]. Haitner and Holesntein [HH09] rule out fully-blackbox constructions of fully-KDM-secure schemes from trapdoor permutations, by giving a family F of $poly(n)$-wise independent hash functions for any proposed construction (G, E, D) and showing that breaking the F-KDM security of (G, E, D) cannot be reduced to inverting trapdoor permutations.[4] One may wonder whether this combined with Applebaum's result [App14], which shows how to amplify projection security into F-KDM security (for any F), already separates projection security from trapdoor permutations. This, however, is not the case. Roughly speaking, in [HH09] the $poly(n)$ parameter of the derived KDM function family F (for which they show that the F-KDM security of the proposed scheme (G, E, D) does not reduce to trapdoor permutations) depends on the complexity of E. On the other hand, under Applebaum's amplification the target function family F' (for which we want to have F'-KDM security) is first fixed, and based on F' an F'-KDM secure scheme is obtained whose, in particular, encryption function's complexity grow by that of F'. This subtle gap does not seem that can be filled at this moment.

Other related work. The question of what "general" assumptions may be used to obtain KDM security is addressed in [HKS16], where it is shown that projection-secure public-key bit encryption (PKE) can be built from any CPA-secure PKE with some additional properties. The power of circular-secure encryption is addressed in [HK15], where it is shown that in combination with the so-called reproducibility property, bit circular security implies the existence of powerful primitives including correlation-secure trapdoor functions [RS10], CCA2-secure encryption and deterministic encryption.

1.2 Blackbox separations and non-blackbox techniques

Blackbox separations. To prove (in a mathematical sense) that primitive P_1 implies primitive P_2 (or P_2 *reduce* to P_1) it suffices to give a construction algorithm C and a security-proof algorithm Red in such a way

[4] The separation results of [HH09] is for the single key case.

that $C(M_1)$ gives an efficient implementation of P_2 whenever M_1 is an efficient implementation of P_1, and that *Red* reduces any efficient attack A_2 against $C(M_1)$ to one against M_1. Most cryptographic reductions are *black-box* in that C only uses M_1 as an *oracle* (denoted C^{M_1}) without assuming anything beyond the input-output correctness of M_1; moreover, *Red* reduces any attack A_2 against C^{M_1} to one against M_1, while treating both M_1 and A_2 as oracles. Such a pair (C, Red) for a cryptographic implication is referred to as a *fully-black-box reduction* [RTV04].

Impagliazzo and Rudich [IR89] were the first to formalize a model in which to show that certain cryptographic implications cannot be proved in a fully-black-box way. In particular, they showed a fully-black-box separation between secret-key agreement protocols and one-way functions.[5] The framework of [IR89] was subsequently used (and extended) to separate many cryptographic primitives from many others in a fully-black-box way, e.g., [IR89,Sim98,GKM+00,GMR01,BPR+08,HH09,Vah10,RS10,MP12]. Some subsequent works showed separations (between certain primitives) w.r.t. reductions under which only the security proof is required to be black-box (and the construction could be non-black-box) [Pas11,GW11,Pas13].

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 [CDSMW08,IKLP06] as examples of blackbox constructions that replaced their earlier non-blackbox counterparts [PSV06,GMW87]. Classical examples of non-blackbox constructions with no known corresponding blackbox results include [NY90,DDN91], giving non-blackbox constructions of CCA1- and CCA2-secure encryption from doubly enhanced trapdoor permutations [Gol11,GR13]. 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.* [GMM07], ruling out *sheilding* blackbox constructions of CCA1-secure encryption from CPA-secure encryption, and that of Myers and Shelat [MS09] proving equivalence of one-bit and many-bit CCA2 secure encryption. Finally, we mention that the work of Mahmoody and Pass [MP12] 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.

More on non-blackbox techniques. There are a number of cryptographic primitives that as tools naturally enable non-blackbox techniques. These include certain technqiues based on FHE and more recently introduced primitives such as witness encryption [GGSW13], functional encryption [BSW11] and indistinguishability obfuscation [BGI+12,GGH+16], as well as techniques based on generic zero-knowledge proofs [GMW91,FLS90]. For example, almost all reductions proved in [GGSW13,SW14] are non-black-box. Some works explore the limitations of some of these non-black-box techniques, by showing how to capture these techniques inside fully-black-box models [BKSY11,AS,AS16]. For example, the results of Asharov and Segev [AS,AS16] only rule out fully-black-box reductions to their base primitives, but their base primitives are designed in such a way that the derived results also rule out common non-black-box techniques associated with the use of indistinguishability obfuscation (e.g., the punctured programming technique [SW14]).

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* (or *validity*) condition.

[5] We remark that the results of [IR89] also rule out more general reductions, including relativizing reductions and so-called semi-black-box reductions (following the terminology of [RTV04]). For the latter, they relied on the assumption $\mathbf{P} \neq \mathbf{NP}$, which later became unconditional by Reingold *et al.* [RTV04].

(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.

Notions of security. 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 single-bit PKE with seed space $\{0,1\}^n$. Let*

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

where $S_i \leftarrow \{0,1\}^n$ and $(SK_i, PK_i) = G(S_i)$, for $1 \le i \le t$. 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 $1/2$.*
- *$\mathcal{E}$ is t-seed circular secure if no PPT adversary $\mathcal{A}$ can break the t-seed circular security of $\mathcal{E}$. Similarly, we can define t-circular security.*

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 implications: (a) CPA security $\Leftrightarrow$ 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 [RTV04,BBF13] for more general notions of blackbox reductions.

Definition 2. *A* fully-blackbox reduction of *P-secure (e.g., circular-secure) PKE to Q-secure (e.g., CPA-secure) PKE 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. The output of the distribution is an

ideally secure single-bit PKE $(\mathbf{g}, \mathbf{e}, \mathbf{d})$ with certain input/output size constraints, as well as two auxiliary oracles $\mathbf{u}$ and $\mathbf{w}$.

We largely follow the notational style of [GMM07]. 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 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 the 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 sometimes write $O(qry) = u$. If Que is a set of 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 of all, note that 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 of 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 [GMR01], where for every candidate construction (G, E, D), we will define a weakening oracle $\mathbf{T}$ 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 define an oracle $\mathbf{T}$ such that

(I) $\mathbf{T}$ is helpful in breaking the (alleged) 1-seed circular security of $\mathcal{E}^{\mathbf{O}}$. That is, there exists a PPT adversary $\mathcal{B}$, where $\mathcal{B}^{\mathbf{O},\mathbf{T}}$ breaks the 1-seed circular security of $(G^{\mathbf{O}}, E^{\mathbf{O}}, D^{\mathbf{O}})$.
(II) $\mathbf{T}$ is not helpful in breaking the CPA security of $\mathbf{O}$. That is, no PPT oracle adversary $\mathcal{A}^{\mathbf{O},\mathbf{T}}$ can break the CPA security of $\mathbf{O}$.

To define $\mathbf{T}$, 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 $E^{\mathbf{O}}(PK, \cdot)$, $\mathbf{T}$ 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 at least two problems with the above naive 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. Essentially, the formal way to ensure this is to design $\mathbf{T}$ in such a way that responses to $\mathbf{T}$-queries of $\mathcal{A}$ can be simulated using access to some "safe" oracles.

Our approach starts by resolving the first problem, using an idea from [GMR01] (also used in some subsequent works [GMM07,Vah10]): 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. That is, $\mathbf{T}$ 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 be defined in any arbitrary way. Thus, we need to ensure that for any plaintext bit b with high probability over a random R:

$$E^{\mathbf{O}}(PK, b; R) = E^{\widetilde{\mathbf{O}}}(PK, b; R). \tag{1}$$

This will ensure that, if $(PK, C_1, \ldots, C_n)$ were "honestly" generated, then $D^{\widetilde{\mathbf{O}}}(SK', C_1 \ldots C_n)$ with high probability will be the real output. This will in turn show that $\mathbf{T}$ is useful in breaking the 1-seed circular security of $(G^{\mathbf{O}}, E^{\mathbf{O}}, D^{\mathbf{O}})$. It remains to show how to define such an oracle $\widetilde{\mathbf{O}}$, and how to ensure that queries to the resulting oracle $\mathbf{T}$ by a CPA adversary $\mathcal{A}^{\mathbf{O},\mathbf{T}}(pk, c)$ can be simulated safely.

Specifically, we construct $\widetilde{\mathbf{O}}$ by *super-imposing* a poly number of query/response pairs Q, 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 in such a way that $G^{\mathsf{Q}} = (*, PK)$. Then, we super-impose Q on $\mathbf{O}$ to obtain $\widetilde{\mathbf{O}}$. The superimposed oracle $\widetilde{\mathbf{O}}$ must agree with Q, must be a valid PKE oracle, and must also agree with the original $\mathbf{O}$ as much as possible. In particular, this last requirement will ensure that Equation 1 above is satisfied.[6]

To ensure that $\mathbf{T}$ is simulatable, 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

[6] In general, to ensure that Equation 1 will hold, we will also need to ensure that Q in turn agrees with "heavy" queries to $\mathbf{O}$ made by $\mathcal{E}$. However, this will not be necessary for the simple class of constructions that we will analyze in the rest of this section.

$\mathcal{A}$'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 $\mathbf{T}$ does not refuse to decrypt too often, making it still useful for breaking the seed circular security of $(G^{\mathbf{O}}, E^{\mathbf{O}}, D^{\mathbf{O}})$.

To summarize, the oracle $\mathbf{T}(PK, C_1, \ldots, C_n)$ will perform a decryption $D^{\widetilde{\mathbf{O}}}(SK', C_1 \ldots C_n)$ and will release the result if some condition holds. Moreover, the oracle $\widetilde{\mathbf{O}}$ is obtained by superimposing a set Q of query/response pairs on $\mathbf{O}$, and we have $G^{\mathsf{Q}} = (SK', PK)$.

Concrete overview. We now give a concrete overview of the above approach for a simple class of constructions, those with oracle access of the form $(G^{\mathbf{g}}, E^{\mathbf{e}}, D^{\mathbf{d}})$. We call these *type-1* constructions.

Recall that the first step involved in the computation of $\mathbf{T}(PK, C_1, \ldots, C_n)$ is to superimpose a set Q of query/response pairs (where $G^{\mathsf{Q}} = (*, PK)$) on $\mathbf{O}$. Since we are dealing with type-1 constructions this set Q will only have $\mathfrak{g}$-type queries. Thus, we start by defining the task of super-imposing a set of $\mathfrak{g}$-type query/response pairs on an oracle $(\mathbf{g}, \mathbf{e}, \mathbf{d})$. (Recall that the notation $\mathfrak{g}$ denotes symbolic queries. See Section 2.)

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*

$$\widetilde{\mathbf{g}}(sk) = \begin{cases} \mathbf{g}(sk) & \text{if } sk \notin \{sk_1, \ldots, sk_w\} \\ pk_i & \text{if } sk = sk_i \text{ for some } 1 \leq i \leq w \end{cases} \tag{2}$$

$\widetilde{\mathbf{d}}(sk, c)$ *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 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 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 frequently in the paper. Informally speaking, it shows one particular situation where queries to $\widetilde{\mathbf{d}}$ (where $\widetilde{\mathbf{d}}$ was defined 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 the forthcoming statement.) Let* Q_{s} *be a set of query/response pairs meeting the condition of Definition 5 and* $(\widetilde{\mathbf{g}}, \widetilde{\mathbf{d}}) = KeyImpose(\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathsf{Q}_{\mathsf{s}})$*. Assuming* $(\langle \mathfrak{g}, * \rangle, pk) \notin \mathsf{Q}_{\mathsf{s}}$*, then* $\mathcal{B}^{\mathbf{g},\mathbf{e},\mathbf{d},\mathbf{u},\mathbf{w}}(pk, \ldots)$*, by having* 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) $(\langle \mathfrak{g}, sk' \rangle, *) \notin \mathsf{Q}_{\mathsf{s}}$ or (ii) for some $pk' \neq pk$, $(\langle \mathfrak{g}, sk' \rangle, pk') \in \mathsf{Q}_{\mathsf{s}}$. 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 (type-1 or otherwise).

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.[7]

[7] 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.

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 presentation. Indeed, we are not aware of any construction that does not satisfy this assumption.

Separation for type-1 constructions. We first describe the oracle $\mathbf{T}$, defined w.r.t. a fixed $(\mathbf{g}, \mathbf{e}, \mathbf{d})$ and a fixed type-1 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: $(1^n, PK, C_1, \ldots, C_n)$
Operations:

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 briefly discuss why $\mathbf{T}$ provides the "desired" properties. We remind the reader that the presentation in the rest of this section is largely informal.

4.1.1 T does not break the CPA security of a random $(\mathbf{g}, \mathbf{e}, \mathbf{d})$.

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 polynomial. 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 $1 \leq i \leq t$.

The proof of the above lemma is based on simple probability arguments and is given in Section A of the appendix.

Fix a Ψ-valid oracle $(\mathbf{O}, \mathbf{u}, \mathbf{w})$. As stated earlier, our goal is to 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 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. We stress that $\mathcal{B}$ will make exponential offline computations during the simulation, but will make a poly-related number of queries. As we will show below, the adversary $\mathcal{B}$ will either be able to perfectly simulate $\mathcal{A}$, or will learn its challenge secret key, $\mathbf{g}^{-1}(pk)$, along the way.

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. Recall that SK' and $\mathsf{Q_s}$ are sampled in such a way that $(SK', PK) = G^{\mathsf{Q_s}}(S')$, for some S'. The adversary $\mathcal{B}$ is able to do these samplings since during these 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) $(\langle \mathfrak{g}, * \rangle, pk) \notin \mathsf{Q_s}$: 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 letting $S_{out} = D^{\widetilde{\mathbf{d}}}(SK', C_1, \ldots, C_{n_1})$, then $\mathcal{B}$ will perform 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) $(\langle \mathfrak{g}, * \rangle, pk) \in \mathsf{Q_s}$: 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., the challenge secret key of $\mathcal{B}$, which is $\mathbf{g}^{-1}(pk)$, will be 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 will return $\perp$, 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) $(\langle \mathfrak{g}, sk' \rangle, *) \notin \mathsf{Q_s}$; or

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

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

(iv) $(\langle \mathfrak{g}, sk' \rangle, pk) \in \mathsf{Q_s}$ 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) $\mathcal{B}$ will continue 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 $\mathcal{B}$ will reply 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 $(\langle \mathfrak{g}, sk' \rangle, pk) \in \mathsf{Q_s}$) $\mathcal{B}$ will reply to qu' with b', making it consistent with the previous reply on this branch. Thus, these two branches will result in two strings S_0, S_1 satisfying the claim.

We have shown that $\mathcal{B}^{\mathbf{O},\mathbf{u},\mathbf{w}}(1^n, pk, c)$ can either fully simulate $\mathcal{A}^{\mathbf{O},\mathbf{T}}(1^n, pk, c)$ or will learn its challenge secret key $\mathbf{g}^{-1}(pk)$ along the way. Moreover, $\mathcal{B}$ makes a poly related number of queries.

Conclusion 2. *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* $m \times poly(n) << 2^{n/4}$*) 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 1-seed circular security of (G, E, D). We will now show that the oracle $\mathbf{T}$ is indeed useful in breaking the purported 1-seed circular security of $(G^{\mathbf{g}}, E^{\mathbf{e}}, D^{\mathbf{d}})$. We first give some intuition about the proof. Let $(PK, C_1, \ldots, C_n)$ be an honestly-generated input to $\mathbf{T}$: that is,

$$S \leftarrow \{0,1\}^n,\ (SK, PK) = G^{\mathbf{g}}(S) \text{ and } (C_1, \ldots, C_n) \leftarrow E^{\mathbf{e}}(PK, S). \tag{3}$$

Now consider the execution of $\mathbf{T}(1^n, PK, C_1, \ldots, C_n)$. It is easy to see that it will always hold $S_{out} = S$, where S_{out} is the string decoded in Step 3 of $\mathbf{T}$'s computation. Thus, we are left to show that the bad

condition specified in Step 3, which makes $\mathbf{T}$ return $\perp$, will hold only with small probability. We will show that whenever this bad condition holds, we are able to produce a valid public key $pk \in \mathbf{g}(\{0,1\}^n)$ without calling $\mathbf{g}$ on pk's pre-image. The following lemma will show that this probability is exponentially 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*

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

The proof of the above lemma is again based on a simple probability argument, and is given in Section A of the appendix. We now formally state and prove the main claim regarding the usefulness of $\mathbf{T}$.

Claim 3. *The oracle $\mathbf{T}$ is useful if used honestly: assuming*

$$\mathcal{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{u}, \mathbf{w}) \leftarrow \Psi,\ S \leftarrow \{0,1\}^n,\ (SK, PK) = G^{\mathbf{g}}(S) \text{ and } (C_1, \ldots, C_n) \leftarrow E^{\mathbf{e}}(PK, S) \tag{4}$$

the probability that $\mathbf{T}(1^n, PK, C_1, \ldots, C_n)$ does not return S is at most $\frac{1}{2^{2n}}$.

Proof. 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

$$(SK', PK) = G^{\mathsf{Q_s}}(1^n, S'),\quad (\widetilde{\mathbf{g}}, \widetilde{\mathbf{d}}) = KeyImpose(\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathsf{Q_s}),\quad S_{out} = D^{\widetilde{\mathbf{d}}}(SK', C_1 \ldots C_n). \tag{5}$$

Also, recall the way $S, PK, C_1, \ldots, C_n$ are chosen in the claim (Equation 4).

We first claim $S_{out} = S$. This follows from the following three statements:

(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 (see the paragraph after Definition 5);
(c) $(SK', PK) = G^{\widetilde{\mathbf{g}}}(1^n; S')$, which follows since $(SK', PK) = G^{\mathsf{Q_s}}(1^n, S')$ and $\widetilde{\mathbf{g}}$ 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 $\Pr[Fail] \leq \frac{1}{2^{2n}}$, proving the claim. We first start with some intuition behind the forgery.

Recall that $S_{out} = S$. By definition, $Fail$ occurs if there exists pk such that

(I) $\mathbf{w}(pk) = \top$;
(II) pk is embedded in $\mathsf{Q_s}$, i.e., $(\langle \mathfrak{g}, * \rangle, pk) \in \mathsf{Q_s}$; and
(III) the query/response $(\langle \mathbf{g}, * \rangle, pk)$ does not show up during $G^{\mathbf{g}}(1^n, S)$.

Now the forgery is done by simply returning pk, which is indeed a valid forgery because $\mathsf{Q_s}$ is produced based on PK in offline mode, and PK is produced as $(SK, PK) = G^{\mathbf{g}}(1^n, S)$, guaranteeing that pk was never previously outputted by $\mathbf{g}$ during the formation of $\mathsf{Q_s}$. The only thing left is to prove that the forged pk is indeed in $\{0,1\}^{5n}$. (This is required by Lemma 2.) 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.

The adversary $\mathcal{A}^{\mathcal{O}}(1^n)$ proceed as follows:

1. *Online mode:* $\mathcal{A}^{\mathcal{O}}(1^n)$ generates $S \leftarrow \{0,1\}^n$ and $(SK, PK) = G^{\mathbf{g}}(S)$;
2. *Offline mode* $\mathcal{A}^{\mathcal{O}}(1^n)$ 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 the execution of $G^{\mathbf{g}}(S)$), for all pk such that $(\langle \mathfrak{g}, * \rangle, pk) \in \mathsf{Q_s}$ 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. $\square$

We can now, using standard techniques, combine Conclusion 2 and Claim 3 to rule out all fully-blackbox type-1 constructions of 1-seed circular secure bit encryption from CPA-secure encryption.

We conclude this subsection with the following remark.

Remark 4. *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 for checking whether the given plaintext is the seed of the given public key. When discussing the general separation result in Section 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

The reader may optionally read this subsection right before reading Section 6, and move on to Section 5.

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 our techniques for overcoming the difficulties in proving a general separation result. Again, we focus on type-1 constructions. Since in this case the seed in the constructed scheme is encrypted as a whole we denote a seed encryption as $C \leftarrow E(PK, S)$ (as opposed to $(C_1, \ldots, C_n) \leftarrow E(PK, S)$).

Fix the proposed full-length 2-seed circular secure 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 the bit encryption scheme $(\mathbf{g}, \mathbf{e}, \mathbf{d})$.

For this new setting, we cannot use the previous approach, mainly because the analysis there for showing that access to the oracle $\mathbf{T}$ of a CPA adversary $\mathcal{A}^{\mathbf{O},\mathbf{T}}(1^n, pk, c)$ is simulatable by a CCA-valid adversary $\mathcal{B}^{\mathbf{O},\mathbf{u},\mathbf{w}}(1^n, pk, c)$ against $(\mathbf{g}, \mathbf{e}, \mathbf{d})$ (Subsection 4.1.1) heavily relies on the fact that only one challenge ciphertext is present (which is c), 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, and hence a two-branch style technique will result in an exponential number of branches. Thus, we need some new ideas for the oracle $\mathbf{T_2}$. We describe the new ideas below. The construction of $\mathbf{T_2}$ will also use some of the previous ideas, which we will only briefly sketch.

Main ideas behind $\mathbf{T_2}$. The oracle $\mathbf{T_2}$ will accept inputs of the form (PK_0, PK_1, C_0, C_1), where purportedly C_i is an encryption of PK_{1-i}'s seed under $E(PK_i, \cdot)$. Then, $\mathbf{T_2}$ will decrypt C_0 and C_1 relative to, respectively, oracles $\widetilde{\mathbf{O_0}}$ and $\widetilde{\mathbf{O_1}}$, obtained by superimposing two sampled sets Q_s^0 and Q_s^1, meeting a certain condition, on $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d})$. Specifically, $\mathbf{T_2}$ samples two sets of query/response pairs Q_s^0 and Q_s^1 in such a way that for $i = 0, 1$ both conditions below hold:

(a) $G^{\mathsf{Q}_\mathsf{s}^\mathsf{i}} = (SK_i', PK_i)$ for some SK_i';
(b) the embedded public keys in Q_s^0 and Q_s^1 are disjoint. That is, for no pk': $(\langle \mathfrak{g}, * \rangle, pk') \in \mathsf{Q}_\mathsf{s}^0$ and $(\langle \mathfrak{g}, * \rangle, pk') \in \mathsf{Q}_\mathsf{s}^1$.

Next, $\mathbf{T_2}$ forms $\widetilde{\mathbf{O_0}} = (\widetilde{\mathbf{g}_0}, \mathbf{e}, \widetilde{\mathbf{d}_0})$ and $\widetilde{\mathbf{O_1}} = (\widetilde{\mathbf{g}_1}, \mathbf{e}, \widetilde{\mathbf{d}_1})$ as

$$(\widetilde{\mathbf{g}_i}, \widetilde{\mathbf{d}_i}) = KeyImpose(\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathsf{Q}_\mathsf{s}^\mathsf{i}), \text{ for } i = 0, 1.$$

Then, $\mathbf{T_2}$ will perform the decryptions

$$S_{out}^0 = D^{\widetilde{\mathbf{d}_1}}(SK_1', C_1)$$

and

$$S^1_{out} = D^{\widetilde{\mathbf{d}_0}}(SK'_0, C_0).$$

Finally, $\mathbf{T_2}$ returns S^0_{out} if the following condition, that we call $Safe$, holds:

$Safe$: 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 a 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 that pk is $\mathcal{B}$'s challenge public key.) Let $i \in \{0, 1\}$ be the index that $\mathcal{B}$ can decrypt C_i, and hence retrieve S^{1-i}_{out}. We now consider two cases:

- $(\langle \mathfrak{g}, * \rangle, pk) \notin \mathsf{Q}^{\mathsf{1-i}}_{\mathsf{s}}$: then $\mathcal{B}$ can also decrypt C_{1-i} to retrieve S^i_{out}. Now having both S^0_{out} and S^1_{out}, $\mathcal{B}$ can easily perform the rest of the execution of $\mathbf{T_2}$.
- $(\langle \mathfrak{g}, * \rangle, pk) \in \mathsf{Q}^{\mathsf{1-i}}_{\mathsf{s}}$: then Condition $Safe$ will guarantee that either the answer to the underlying query to $\mathbf{T_2}$ is $\perp$, or $\mathcal{B}$ will learn its challenge secret key (i.e., $\mathbf{g}^{-1}(pk)$) along the execution of $G^{\mathbf{g}}(S^{1-i}_{out})$.

The check (b) however may make the oracle $\mathbf{T_2}$ too weak to break the 2-seed circular security of $(G^{\mathbf{g}}, E^{\mathbf{e}}, D^{\mathbf{d}})$. In particular, if there are pk's that occur quite frequently 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, due to Check (b). To give some intuition, consider a contrived construction $(G^{\mathbf{g}}, E^{\mathbf{e}}, D^{\mathbf{d}})$, where $G^{\mathbf{g}}(1^n)$, regardless of its given seed, first calls $\mathbf{g}(0^n)$ to get some pk', and at the end outputs something like $PK = (pk', *)$. Whenever sampling $\mathsf{Q_s}$ satisfying $G^{\mathsf{Q_s}} = (*, PK)$, the key pk' will be embedded in $\mathsf{Q_s}$.

We overcome the above problem as follows. We first sample a large number of executions of $G^{\mathbf{O}}(1^n)$, record all the query/response pairs and make $\mathsf{Q}^0_{\mathsf{s}}$ and $\mathsf{Q}^1_{\mathsf{s}}$ be consistent with this information. The idea is that if there are "frequent" pk' in the sense above, they should be collected with high probability during these executions. We will then require $\mathsf{Q_0}$ and $\mathsf{Q_1}$ to be disjoint only on public keys that did not turn up as a result of these executions.

Lastly, we note that this additional step does not ruin our preceding technique for simulating responses to oracle queries to $\mathbf{T_2}$, since if the challenge public key pk turns up during any of these executions then we will have learned the challenge secret key; otherwise, the above simulation strategy goes through.

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 $(\mathbf{g}'_{\mathsf{i}}, S'_i)$ 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 $\mathsf{QPub_i}$ have any pk such that $\mathbf{w}(pk) = \top$ 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
$$S^1_{out} = D^{\widetilde{\mathbf{d}_0}}(SK'_0, C_0) \quad \text{and} \quad S^0_{out} = D^{\widetilde{\mathbf{d}_1}}(SK'_1, C_1).$$
Return S^0_{out} if the following condition holds for both $i = 0, 1$, and return $\perp$, otherwise: For all $pk \in \mathsf{QPub_i} \setminus \mathsf{FreqPub}$ 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 $\mathbf{O}$. For any adversary $\mathcal{A}^{\mathbf{O},\mathbf{T_2}}(1^n, pk, c_1, \ldots, c_n, c)$, against the 1-seed circular-security of $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d})$, we show that it may 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
$$Tqu = \langle \mathbf{T_2}, (1^n, PK_0, PK_1, C_0, C_1) \rangle$$
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 $i \in \{0, 1\}$ $pk \notin \mathsf{QPub_i}$ since otherwise by Line 3 of $\mathbf{T_2}$ 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, $\mathbf{B}$ will either find its challenge secret key or will correctly discern 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 will show that the probability that $\mathbf{T_2}(1^n, PK_0, PK_1, C_0, C_1)$ does not return S_0 is small. (Formally, it can be bounded by any inverse polynomial by choosing the value of ϱ in step 1 of $\mathbf{T_2}$'s computation accordingly.)

First, as in the corresponding proof in Subsection 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 the events $Ev1$ and $Ev2$, defined as follows:
$$Ev1 = (\mathsf{QPub_0} \cap \mathsf{QPub_1}) \setminus \mathsf{FreqPub} \neq \emptyset$$
and
$$Ev2 = ((\mathsf{QPub_0} \not\subseteq \mathsf{RealPub_0} \cup \mathsf{FreqPub}) \vee (\mathsf{QPub_1} \not\subseteq \mathsf{RealPub_1} \cup \mathsf{FreqPub})), \tag{6}$$
where
$$\mathsf{RealPub_i} = \{pk \mid \text{ the query/response } (\langle \mathbf{g}, * \rangle, pk) \text{ occurs during } G^{\mathbf{g}}(S_i)\}.$$
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 Subsection 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 have also occurred 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 case

In this section we describe the weakening oracle $\mathbf{T}$ for general constructions of 1-seed circular secure bit encryption, those in which access to the base primitive oracle is unrestricted. We will then 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. We did not have this problem for type-1 constructions analyzed in Section 4.1 because in those constructions the algorithms G and E have access to different sub-oracles.

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 will then force the sampled set Q to be *consistent* with Freq. Finally, we will show how to superimpose Q on $\mathbf{O}$ to obtain $\widetilde{\mathbf{O}}$.

5.1 Tools for defining the oracle T

Fix the purported 1-seed circular secure bit encryption construction (G, E, D) throughout this section. We will now give an assumption to make our analysis simpler and then give definitions formalizing the steps sketched above. We will 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 a query $\langle \mathbf{d}, (sk, c) \rangle$: 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.

We define another procedure that will be used by the oracle $\mathbf{T}$. Given a public key PK of the constructed scheme and a set Freq of query/response pairs obtained based on $(\mathbf{g}, \mathbf{e}, \mathbf{d})$ we define a procedure that samples a matching secret key SK' for PK with respect to a new oracle $(\mathbf{g}', \mathbf{e}', \mathbf{d}')$ that agrees with Freq. The procedure then returns all the underlying queries made to produce (SK', PK).

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*

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

  scheme. 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

$$\mathbf{e}_{imp}(pk, b, r) = \begin{cases} c_i & \text{if } (pk, b, r) = (pk_i, b_i, r_i), \text{ for some } 1 \le i \le p \\ \hat{c} & \text{if } (pk, \mathbf{e}(pk, b, r)) \in \mathsf{W} \\ \mathbf{e}(pk, b, r) & \text{otherwise} \end{cases} \tag{7}$$

  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.

$$\mathbf{d}_{imp}(sk, c) = \begin{cases} b_i & \text{if } \mathbf{g}(sk) = pk_i \text{ and } c = c_i \text{ for some } 1 \le i \le p \\ \mathbf{d}(sk, c) & \text{otherwise} \end{cases} \tag{8}$$

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 $\hat{c}$ such that $(pk, \hat{c}) \notin \mathsf{W} \cup \mathsf{W}'$. The requirement $(pk, \hat{c}) \notin \mathsf{W}'$ is stronger than necessary, but will simplify some proofs. Note that $\mathbf{e}_{imp}$ is not necessarily injective.

5.2 Description of the oracle 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
$$(SK', \mathsf{Q_s}, \mathsf{Q_c}) \leftarrow ConsOrc(PK, \mathsf{Freq}). \tag{9}$$
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.

Remark about Step 1 of T. This step comes with a big polynomial, $n^{23\vartheta}$, which essentially dictates how many times $E^{\mathbf{O}}(PK, 01)$ will be executed. We chose such a big polynomial just for the convenience of having much slackness in the bounds shown in security proofs. This polynomial can be made smaller, but the proofs will become more tedious. In fact, since the aim is to prove separations, the exact polynomial efficiency of the separation oracle becomes less important.

5.3 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 5. *It holds that*

$$\Pr_{\mathbf{Env},\mathbf{T}}[\mathbf{T}(PK, C_1, \ldots, C_n) = S] \geq 1 - \frac{1}{n^5}, \tag{10}$$

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 that the probability that $S_{out} = S$ and $\mathbf{T}(PK, C_1, \ldots, C_n) = \perp$ 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* $\mathbf{T}(PK, C_1, \ldots, C_n)$.

We now show how to prove Theorem 5 based on Lemmas 3 and 4. We will then prove Lemma 3 in Subsection 5.4 and Lemma 4 in Subsection 5.5.

Proof (of Theorem 5). Let $Evnt$ be the event that $\mathbf{T}(PK, C_1, \ldots, C_n) \neq S$. We show that

$$\Pr_{\mathbf{Env},\mathbf{T}}[Evnt] \leq \frac{1}{n^5}. \tag{11}$$

We define the following two events:

- $Evnt1$: the event that $D^{\widetilde{\mathbf{O}}}(SK', C_1 \ldots C_n) \neq S$.
- $Evnt2$: the event that

$$(D^{\widetilde{\mathbf{O}}}(SK', C_1 \ldots C_n) = S) \wedge (\mathbf{T}(PK, C_1, \ldots, C_n) = \perp)$$

We claim

$$\Pr_{\mathbf{Env},\mathbf{T}}[Evnt] \leq \Pr_{\mathbf{Env},\mathbf{T}}[Evnt1 \vee Evnt2].$$

Note that by Lemmas 3 and 4 the above claim immediately implies

$$\Pr_{\mathbf{Env},\mathbf{T}}[Evnt] \leq \frac{1}{2n^5} + \frac{1}{2^{2n}} \leq \frac{1}{n^5},$$

as desired. Thus, we show why our claim holds. To this end, we show whenever $Evnt$ holds, also $Evnt1 \vee Evnt2$ holds; or equivalently, if $\overline{Evnt1} \wedge \overline{Evnt2}$ holds, then $\overline{Evnt}$ holds. If $\overline{Evnt1} \wedge \overline{Evnt2}$ holds then

(a) $D^{\widetilde{\mathbf{O}}}(SK', C_1 \dots C_n) = S$; and
(b) $\mathbf{T}(PK, C_1, \dots, C_n) \neq \perp$.

The way in which $\mathbf{T}$ is designed guarantees that if (a) and (b) hold, then $\mathbf{T}(PK, C_1, \dots, C_n) = S$, namely $\overline{Evnt}$ holds. The proof is now complete. □

5.4 Proof of Lemma 3

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

Fact 6. *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) $(SK', PK) \in G^{\widetilde{\mathbf{O}}}(1^n)$.*

Proof. First of all, recall that

$$(\mathbf{e}_{imp}, \mathbf{d}_{imp}) = EncImpose(\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathsf{Q_c})$$
$$(\widetilde{\mathbf{g}}, \widetilde{\mathbf{d}}) = KeyImpose(\mathbf{g}, \mathbf{e}_{imp}, \mathbf{d}_{imp}, \mathsf{Q_s})$$
$$\widetilde{\mathbf{e}} = \mathbf{e}_{imp}.$$

We claim

Claim 7. *$(\mathbf{g}, \mathbf{e}_{imp}, \mathbf{d}_{imp})$ is a valid PKE.*

Note that the above claim immediately implies, by definition of $KeyImpose$, that $(\widetilde{\mathbf{g}}, \mathbf{e}_{imp}, \widetilde{\mathbf{d}})$ is also a valid PKE, or equivalently $\widetilde{\mathbf{O}} = (\widetilde{\mathbf{g}}, \widetilde{\mathbf{e}}, \widetilde{\mathbf{d}})$ is a valid PKE, as desired. The proof of Claim 7 in turn follows again by a simple inspection of the definition of $EncImpose$.

To prove Part (b) note that $\widetilde{\mathbf{O}}$ fully agrees with $\mathsf{Q_s} \cup \mathsf{Q_c}$ and that $(SK', PK) \in G^{\mathsf{Q_s} \cup \mathsf{Q_c}}$ (see Step 2 of $\mathbf{T}$'s computation).

The proof of Fact 6 is now complete. □

Equipped with Fact 6, toward proving Lemma 3 we bound the probability that, for a random R:

$$E^{\mathbf{O}}(PK, S; R) \neq E^{\widetilde{\mathbf{O}}}(PK, S; R) \tag{12}$$

If this probability is small then with high probability $D^{\widetilde{\mathbf{O}}}(SK', C_1 \dots C_n)$ results in S, as desired. We will actually bound a related probability, where S in Equation 12 is replaced with $0^n 1^n$. (Recall that $|S| = n$.) To this end, we need the following lemma. Informally, the following lemma bounds, for any index i, the probability that the two executions $E^{\mathbf{O}}(PK, 0^n 1^n; R)$ and $E^{\widetilde{\mathbf{O}}}(PK, 0^n 1^n; R)$ are different, and that their difference occurs for the first time in the reply to the ith query.

Lemma 5. *Fix $(\mathbf{O}, \mathbf{u}, \mathbf{w}, PK) \in Env(n)$ and let $M = 0^n 1^n$. Let $(qu_1, \dots, qu_{2n^{\vartheta+1}})$ 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) $\Pr\left[(qu_i \textit{ is } \mathbf{g}\textit{- or } \mathbf{e}\textit{-type}) \wedge \left(\forall j < i, \mathbf{O}(qu_j) = \widetilde{\mathbf{O}}(qu_j)\right) \wedge \left(\mathbf{O}(qu_i) \neq \widetilde{\mathbf{O}}(qu_i)\right)\right] \leq \frac{1}{n^{8\vartheta}}$,

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

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

$$(\mathbf{O}, \mathbf{u}, \mathbf{w}, S, PK, C_1, \ldots, C_n) \leftarrow Env(n), \quad (\mathsf{Freq}, SK', \mathsf{Q_s}, \mathsf{Q_c}, \widetilde{\mathbf{O}}) \leftarrow Tvars^{\mathbf{O},\mathbf{u},\mathbf{w}}(PK).$$

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 6 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 $D^{\widetilde{\mathbf{O}}}(SK', C_1 \ldots C_n) = S$.

We thus show how to bound $\beta(n)$. Let QS' denote the set of all queries asked during a random execution of $E^{\mathbf{O}}(PK, M)$, where $M = 0^n 1^n$. We claim

$$\beta(n) \leq \beta'(n) \stackrel{\text{def}}{=} \Pr[\exists qu \in \mathsf{QS}' \colon \mathbf{O}(qu) \neq \widetilde{\mathbf{O}}(qu)]. \tag{13}$$

Equation 13 holds because: if S has k 0's, then QS is identically distributed to the set of queries asked during a random execution of $E^{\mathbf{O}}(PK, 0^k 1^{n-k})$. Moreover, since $k \leq n$, the probability that during a random execution of $E^{\mathbf{O}}(PK, 0^k 1^{n-k})$ 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 prove Lemma 5, starting with Part (A). To this end, we need the following lemma which describes when a query's responses differ between $\mathbf{g}$ and $\widetilde{\mathbf{g}}$ or between $\mathbf{e}$ and $\widetilde{\mathbf{e}}$.

Lemma 6. *For any*

$$(\mathbf{O}, \mathbf{u}, \mathbf{w}, PK) \in Env(n) \textit{ and } (\mathsf{Freq}, SK', \mathsf{Q_s}, \mathsf{Q_c}, \widetilde{\mathbf{O}}) \in Tvars^{\mathcal{O}}(PK) \tag{14}$$

all the following conditions hold:

1. $\mathbf{O}$ and $\widetilde{\mathbf{O}}$ both agree with Freq. *That is, if* $(\langle \mathfrak{g}, q \rangle, ans) \in \mathsf{Freq}$, *then* $\mathbf{g}(q) = \widetilde{\mathbf{g}}(q) = ans$. *Similarly, if* $(\langle \mathfrak{e}, q \rangle, ans) \in \mathsf{Freq}$, *then* $\mathbf{e}(q) = \widetilde{\mathbf{e}}(q) = ans$.
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* $\mathbf{u}(pk, c) = (b, r)$.

Proof. First, recall that

$$(\mathbf{e}_{imp}, \mathbf{d}_{imp}) = EncImpose(\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathsf{Q_c})$$
$$(\widetilde{\mathbf{g}}, \widetilde{\mathbf{d}}) = KeyImpose(\mathbf{g}, \mathbf{e}_{imp}, \mathbf{d}_{imp}, \mathsf{Q_s})$$
$$\widetilde{\mathbf{e}} = \mathbf{e}_{imp}.$$

We now prove all parts of the lemma.

- (Part 1) We prove the statement for each query type separately.
  - Suppose $(\langle \mathfrak{g}, sk\rangle, pk) \in \mathsf{Freq}$ for some sk and pk. Thus, $\mathbf{g}(sk) = pk$. Moreover, since $\mathsf{Q_s}$ should agree with Freq we have either $(\langle \mathfrak{g}, sk\rangle, pk) \in \mathsf{Q_s}$ or $(\langle \mathfrak{g}, sk\rangle, *) \notin \mathsf{Q_s}$. Both cases imply $\mathbf{g}(sk) = \widetilde{\mathbf{g}}(sk)$.
  - Suppose $(\langle \mathfrak{e}, (pk, b, r)\rangle, c) \in \mathsf{Freq}$, for some pk, b, r and c. Thus, $\mathbf{e}(pk, b, r) = c$. Since $\mathsf{Q_c}$ is generated based on an oracle $(\mathbf{g}', \mathbf{e}', \mathbf{d}')$ that is consistent with Freq and that is also Ψ-valid (in particular, $\mathbf{e}'$ is injective), we have either $(\langle \mathfrak{e}, (pk, b, r)\rangle, c) \in \mathsf{Q_c}$ or $(\langle \mathfrak{e}, (pk, *, *)\rangle, c) \notin \mathsf{Q_c}$. Both these conditions imply $\mathbf{e}_{imp}(pk, b, r) = \mathbf{e}(pk, b, r)$. The result now follows by noting that $\widetilde{\mathbf{e}} = \mathbf{e}_{imp}$.
- (Part 2) Straightforward.
- (Part 3) Straightforward by noting that $\widetilde{\mathbf{e}} = \mathbf{e}_{imp}$ and applying the definition of $\mathbf{e}_{imp}$.

□

We also need a version of Chernoff-Hoeffding bounds [MR95].

Theorem 8. *(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* $x_{av} = (x_1 + \cdots + x_{n^t})/n^t$

$$\Pr[|x_{av} - p| \geq \frac{1}{n^k}] \leq \frac{1}{2^{2n^{t-2k}}}. \tag{15}$$

We are now ready to prove Part (A) of Lemma 5.

Proof (Lemma 5, Part (A)). Fix $(\mathbf{O}, \mathbf{u}, \mathbf{w}, PK)$ as in the lemma and let

$$\mathsf{QS} = \{qu_1, \ldots, qu_{2n^{\vartheta+1}}\} \tag{16}$$

be the set of all queries made during the execution of $E^{\mathbf{O}}(PK, M; R)$. All probabilities, if not otherwise stated, are taken over the variables sampled in the lemma. We prove a stronger result, showing

$$\alpha(n) = \Pr\left[(\exists qu \in \mathsf{QS} \text{ such that } qu \text{ is } \mathbf{g}\text{- or } \mathbf{e}\text{-type}) \wedge \left(\mathbf{O}(qu) \neq \widetilde{\mathbf{O}}(qu)\right)\right] \leq \frac{1}{n^{8\vartheta}}, \tag{17}$$

We first define a set $\mathsf{DifQue_c}$, that is going to contain all $\mathfrak{e}$-type queries that will receive different responses from $\mathbf{e}$ and $\widetilde{\mathbf{e}}$. That is, we want $\mathsf{DifQue_c}$ to be formed in such a way that if $\mathbf{e}(pk, b, r) \neq \widetilde{\mathbf{e}}(pk, b, r)$ then $\langle \mathfrak{e}, (pk, b, r)\rangle \in \mathsf{DifQue_c}$. To this end, we use Lemma 6 to define $\mathsf{DifQue_c}$ as follows:

- for any $(\langle \mathfrak{e}, (pk, b, r)\rangle, c) \in \mathsf{Q_c}$: add $\langle \mathfrak{e}, (pk, b, r)\rangle$ to $\mathsf{DifQue_c}$;
- for any $(\langle \mathfrak{e}, (pk, b, r)\rangle, c) \in \mathsf{Q_c}$, if $\mathbf{u}(pk, c) = (b', r') \neq \perp$, add $\langle \mathfrak{e}, (pk, b', r')\rangle$ to $\mathsf{DifQue_c}$.

By Lemma 6, we will have that if $\mathbf{e}(pk, b, r) \neq \widetilde{\mathbf{e}}(pk, b, r)$ then $\langle \mathfrak{e}, (pk, b, r)\rangle \in \mathsf{DifQue_c}$.

We similarly define a set $\mathsf{DifQue_s}$ that is going to contain all $\mathfrak{g}$-type queries that will receive different responses from $\mathbf{g}$ and $\widetilde{\mathbf{g}}$. To this end, for any $(\langle \mathfrak{g}, sk\rangle, *) \in \mathsf{Q_s}$ add $\langle \mathfrak{g}, sk\rangle$ to the set $\mathsf{DifQue_s}$. Again by Lemma 6, $\mathsf{DifQue_s}$ is a super-set of all $\mathfrak{g}$-type queries responded to differently under $\mathbf{g}$ and $\widetilde{\mathbf{g}}$.

Fix an arbitrary ordering

$$(query_1, \ldots, query_m)$$

on the elements of $\mathsf{DifQue_c} \cup \mathsf{DifQue_s}$, and note that $m \leq 2n^{\vartheta}$. This because $|\mathsf{Q_c} \cup \mathsf{Q_s}| = n^{\vartheta}$ (see Definition 7) and because the second rule for adding elements to $\mathsf{DifQue_c}$ may result in at most n^{ϑ} new elements.

Recall QS and $\alpha(n)$ from Equations 16 and 17. We claim

$$\alpha(n) \leq \beta(n) \stackrel{\text{def}}{=} \Pr\left[\text{for some } 1 \leq i \leq m\colon \langle query_i \rangle \in \mathsf{QS} \ \wedge \ (\langle query_i \rangle, *) \notin \mathsf{Freq}\right]. \tag{18}$$

The reason behind the above equation is that if the oracles $\mathbf{O}$ and $\widetilde{\mathbf{O}}$ disagree on a $\mathbf{g}$ type or $\mathbf{e}$ type query $\langle qu \rangle$, then by Lemma 6 $(\langle qu \rangle, *) \notin \mathsf{Freq}$ and $(\langle qu \rangle, *) \in \mathsf{DifQue_c} \cup \mathsf{DifQue_s}$.

To bound $\beta(n)$ we consider two cases for each of the m queries $query_1, \ldots, query_m \in \mathsf{DifQue_c} \cup \mathsf{DifQue_s}$:

(A) $\langle query_i \rangle$ is a high-probable query during the execution of $E^{\mathbf{O}}(PK, M)$: for this case we will then argue that $query_i$ should have already been collected by Freq.
(B) $\langle query_i \rangle$ is not a high probable query: for this case we will argue that it is unlikely that $\langle query_i \rangle \in \mathsf{QS}$.

We now proceed formally. Call a query $\langle qu \rangle$ *heavy* if for some $b \in \{0, 1\}$:

$$\Pr_R\left[\text{the query } \langle qu \rangle \text{ is asked during } E^{\mathbf{O}}(PK, b; R)\right] \geq \frac{1}{n^{11\vartheta}}. \tag{19}$$

We call $\langle qu \rangle$ *unheavy* if the above probability is strictly less than $\frac{1}{n^{11\vartheta}}$ for both $b = 0$ and $b = 1$.

Recall that $\mathsf{DifQue_c} \cup \mathsf{DifQue_s} = \{query_1, \ldots, query_m\}$. Now defining

$$\epsilon(n) = \Pr\left[\exists \text{ query } \langle qu \rangle \text{ s.t. } \langle qu \rangle \text{ is heavy } \bigwedge(\langle qu \rangle, *) \notin \mathsf{Freq}\right],$$
$$\beta_{1,i}(n) = \Pr\left[\langle query_i \rangle \in \mathsf{QS} \bigwedge(\langle query_i \rangle, *) \notin \mathsf{Freq} \bigwedge \langle query_i \rangle \text{ is unheavy}\right],$$

we have

$$\begin{aligned}
\beta(n) =& \Pr\left[\exists i \in [m]\colon \langle query_i \rangle \in \mathsf{QS} \ \wedge \ (\langle query_i \rangle, *) \notin \mathsf{Freq} \ \wedge \langle query_i \rangle \text{ is heavy}\right] \\
&+ \Pr\left[\exists i \in [m]\colon \langle query_i \rangle \in \mathsf{QS} \ \wedge \ (\langle query_i \rangle, *) \notin \mathsf{Freq} \ \wedge \langle query_i \rangle \text{ is unheavy}\right] \\
&\leq \Pr\left[\exists \text{ query } \langle qu \rangle \text{ s.t. } \langle qu \rangle \text{ is heavy } \ \wedge \ (\langle qu \rangle, *) \notin \mathsf{Freq}\right] + \sum_{1 \leq i \leq m} \beta_{1,i}(n) \\
&\leq \epsilon(n) + \sum_{1 \leq i \leq m} \beta_{1,i}(n).
\end{aligned}$$

We claim

Claim I. $\beta_{1,i}(n) \leq \frac{1}{2n^{10\vartheta}}$, for all $1 \leq i \leq m$.

Claim II. $\epsilon(n)$ is negligible.

Claims I and II imply

$$\beta(n) \leq negl(n) + 2n^{\vartheta} \times \frac{1}{2n^{10\vartheta}} \leq 2 \times \frac{1}{n^{9\vartheta}} \leq \frac{1}{n^{8\vartheta}},$$

as desired. We now prove the two claims.

Proof of Claim I. Recall that QS contains queries made during $E(PK, 0^n 1^n; R)$ for a random R. We have

$$\beta_{1,i}(n) \leq \Pr\left[(\langle query_i \rangle \in \mathsf{QS}) \wedge (\langle query_i \rangle \text{ is unheavy})\right] \leq \Pr\left[\langle query_i \rangle \in \mathsf{QS} \mid \langle query_i \rangle \text{ is unheavy}\right] \tag{20}$$

and thus $\beta_{1,i}(n) \leq \frac{2n}{n^{11\vartheta}} \leq \frac{1}{n^{9\vartheta}}$, as claimed. Note that we crucially used the fact R, which is the randomness used to produce QS, is chosen independently of the randomness used to generate the values of

$$\{query_1, \ldots, query_m\} = \mathsf{DifQue_c} \cup \mathsf{DifQue_s}.$$

The set $\mathsf{DifQue_c} \cup \mathsf{DifQue_s}$ is obtained deterministically from $(\mathbf{O}, \mathbf{u}, \mathbf{w})$ and $\mathsf{Q_s} \cup \mathsf{Q_c}$. Recall that $(\mathbf{O}, \mathbf{u}, \mathbf{w}, PK) \in Env(n)$ was fixed by the theorem. Moreover, we know that the randomness R is chosen independently of the randomness used to generate $(\mathsf{Freq}, SK', \mathsf{Q_s}, \mathsf{Q_c}, \widetilde{\mathbf{O}}) \leftarrow Tvars^{\mathbf{O},\mathbf{u},\mathbf{w}}(PK)$.

Proof of Claim II. The proof consists of two parts. First, we show there are at most a polynomial number of heavy queries (**Claim A**). Next, for any specific heavy query $\langle qu \rangle$ we show that the probability that $(\langle qu \rangle, *) \notin \mathsf{Freq}$ is negligible (**Claim B**). These two imply **Claim II**.

Proof of Claim A. Fix $b \in \{0, 1\}$. We show the number of queries $\langle qu \rangle$ for which Equation 19 holds for the fixed b is at most polynomial. Call a query $\langle qu \rangle$ *i-heavy* if with probability at least $\frac{1}{n^{12\vartheta}}$ the query $\langle qu \rangle$ is the ith query made during the execution of $E^{\mathbf{O}}(PK, b)$. Since $E^{\mathbf{O}}(PK, b)$ makes n^{ϑ} queries (see Assumption 1), if a query $\langle qu \rangle$ is asked with probability at least $\frac{1}{n^{11\vartheta}}$ during $E^{\mathbf{O}}(PK, b)$, then for some $1 \leq i \leq n^{\vartheta}$, $\langle qu \rangle$ is i-heavy. For every i we have at most $n^{12\vartheta}$ i-heavy queries. Thus, for a fixed b we have at most $n^{12\vartheta} \times n^{\vartheta}$ queries for which Equation 19 holds. Thus, we have at most $2 \times n^{13\vartheta}$ heavy queries.

Proof of Claim B. Recall that Freq is formed by collecting all query/response pairs made during $n^{23\vartheta}$ random executions of $E^{\mathbf{O}}(PK, 01)$. Let $\langle qu \rangle$ be a fixed heavy query. For $1 \leq k \leq n^{23\vartheta}$ let $x_k = 1$ if $\langle qu \rangle$ is asked during the kth execution, and $x_k = 0$, otherwise. We know that $x_1, \ldots, x_{n^{23\vartheta}}$ are all identically distributed and independent, and that $p \stackrel{\text{def}}{=} \Pr[x_1 = 1] \geq \frac{1}{n^{11\vartheta}}$. Let $x_{av} = (x_1 + \cdots + x_{n^{23\vartheta}})/n^{23\vartheta}$. We have

$$\Pr[(\langle qu \rangle, *) \notin \mathsf{Freq} \mid \langle qu \rangle \text{ is heavy}] \leq \Pr[|x_{av} - p| \geq p] \leq \Pr[|x_{av} - p| \geq \frac{1}{n^{11\vartheta}}].$$

By Theorem 8 the last probability above is at most $\frac{1}{2^{2n^{23\vartheta - 22\vartheta}}} \leq \frac{1}{2^{2n}}$. □

The proof of Part (B) of Lemma 5 follows similarly to that of Part (A) by first formulating when for an arbitrary (sk, c), $\mathbf{d}(sk, c) \neq \widetilde{\mathbf{d}}(sk, c)$, and then bounding the probability accordingly. See Section B of the appendix.

5.5 Proof of Lemma 4

The main idea that goes into proving the bound of Lemma 4 was already sketched in Subsection 4.1: we show how to forge a public key in the sense of Lemma 2 if the events inside the probability hold.

Proof. Let $\alpha(n)$ be as in the lemma. To bound $\alpha(n)$, suppose $\mathbf{T}(PK, C_1, \ldots, C_n) = \perp$ and also suppose that $D^{\widetilde{\mathbf{O}}}(SK', C_1 \ldots C_n) = S$. In other words, $S_{out} = S$, where S_{out} is the variable defined inside the computation of $\mathbf{T}$. Then by Step 5 of $\mathbf{T}$'s computation it must hold

$$\mathsf{QPub} \not\subseteq \mathsf{EmbedPub} \cup \mathsf{FreqPub}. \tag{21}$$

Thus,

$$\alpha(n) \leq \Pr_{\mathbf{Env},\mathbf{T}}[\mathsf{QPub} \not\subseteq \mathsf{EmbedPub} \cup \mathsf{FreqPub})], \tag{22}$$

where $\mathbf{Env} = (\mathbf{O}, \mathbf{u}, \mathbf{w}, S, PK, C_1, \ldots, C_n) \leftarrow Env(n)$. We show whenever Equation 21 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 Equations 21 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 $\mathsf{Freq} \leftarrow FreqQue^{\mathbf{O},\mathbf{u}}(PK, n^{23\vartheta})$, 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 Equation 22, $\mathcal{B}$'s success probability is at least $\alpha(n)$. Applying Lemma 2 the desired bound for $\alpha(n)$ follows. □

5.6 T does not break the CPA security of the base scheme

Theorem 9. *Suppose $\mathcal{A}$ is a CPA adversary with access to the oracles $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d})$ and $\mathbf{T}$ that makes at most $2^{n/8}$ queries. We have*

$$\Pr_{\mathbf{O},\mathbf{T},b,sk,c}[\mathcal{A}^{\mathbf{g},\mathbf{e},\mathbf{d},\mathbf{T}}(1^n, pk, c) = b] \leq \frac{1}{2} + \frac{1}{2^{n/4}}, \tag{23}$$

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

In order to be able to simulate access to $\mathbf{T}$ we should be able to efficiently simulate certain queries to $\widetilde{\mathbf{O}}$ via oracle access to $(\mathbf{O}, \mathbf{u}, \mathbf{w})$. To this end, we need the following lemma. The proof of the lemma follows mostly by inspection and is given in Section C of the appendix.

Lemma 7. *Fix*

$$(\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{u}, \mathbf{w}, PK) \in Env(n) \text{ and } (\mathsf{Freq}, SK', \mathsf{Q_s}, \mathsf{Q_c}, \widetilde{\mathbf{g}}, \widetilde{\mathbf{e}}, \widetilde{\mathbf{d}}) \in Tvars^{\mathcal{O}}(PK).$$

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

$$\mathsf{W}' = \{(pk_1, \mathbf{e}(pk_1, b_1, r_1)), \ldots, (pk_p, \mathbf{e}(pk_p, b_p, r_p))\}$$

and

$$\mathsf{W} = \{(pk_1, c_1), \ldots, (pk_p, c_p)\}.$$

Note that in the definition of W' by $\mathbf{e}(pk_i, b_i, r_i)$ we mean the actual value $\mathbf{e}(pk_i, b_i, r_i)$, not the notation itself.

All the following conditions hold.

(a) Both $\widetilde{\mathbf{g}}$ and $\widetilde{\mathbf{e}}$ can be computed efficiently (on all points) given access to the 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 the 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 $\widetilde{\mathbf{d}}(sk, c) = b_i$.

We now show how to prove Theorem 9 using Lemma 7. We first start with a sketch of the proof.

Proof sketch of Theorem 9. As in Section 4 the idea is to give an adversary $\mathcal{B}$, such that $\mathcal{B}^{\mathcal{O}}(1^n, pk, c)$ 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 a query of $\mathcal{A}$, where $Tqu = \langle \mathbf{T}, (1^n, PK, C_1, \ldots, C_n)\rangle$. 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 without loss of generality that $pk \notin \mathsf{FreqPub}$, 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, which is done offline, and then 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 $\mathcal{B}$ will perform 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 such that $S_{out} \in \{S_0, S_1\}$. 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 Case (c) of Lemma 7 holds, namely

$$(\langle \mathfrak{g}, sk'\rangle, pk) \in \mathsf{Q_s}, \quad c' = c \text{ and } \quad (pk, c) \notin \mathsf{W} \cup \mathsf{W}'$$

In this case, knowing that the answer to qu is the challenge bit b, $\mathcal{B}$ will start 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 that $\mathcal{B}$ provides a consistent reply on either branch if the same query shows up in the future. The two strings S_0 and S_1 decoded at the end on the two branches will satisfy the above claim. □

We now proceed to give the full proof of Theorem 9. We will also need the following proposition which is an easy implication of Lemma 7.

Proposition 1. *Fix* $(\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{u}, \mathbf{w}, PK) \in Env(n)$ *and* $(\mathsf{Freq}, SK', \mathsf{Q_s}, \mathsf{Q_c}, \widetilde{\mathbf{g}}, \widetilde{\mathbf{e}}, \widetilde{\mathbf{d}}) \in Tvars^{\mathcal{O}}(PK)$. *Let* W *and* W' *be formed as in Lemma 7. For any* (sk, c) *the following holds:*

1. *If* $(\langle \mathfrak{g}, sk\rangle, *) \notin \mathsf{Q_s}$, *then* $\widetilde{\mathbf{d}}(sk, c)$ *can be efficiently computed given access to the oracle* $\mathbf{O}$ *and having* $\mathsf{Q_c}$ *as input.*
2. *If* $(\langle \mathfrak{g}, sk\rangle, pk) \in \mathsf{Q_s}$, *for some* pk, *then* $\widetilde{\mathbf{d}}(sk, c)$ *can be efficiently computed given access to the oracle* $\mathbf{O}$ *and by knowing* $\mathsf{Q_c}$ *and the value of* $\mathbf{u}(pk, c)$.

Proof (of Theorem 9). We prove a stronger result, in which $\mathcal{A}$ has access to $(\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{u}, \mathbf{w})$ and $\mathbf{T}$, and is restricted to be CCA valid. Fix $\mathcal{A}$ to be an adversary for which

$$\Pr_{\mathcal{O},\mathbf{T},b,pk,c}\left[\mathcal{A}^{\mathcal{O},\mathbf{T}}(1^n, pk, c) = b\right] = \frac{1}{2} + \alpha(n), \tag{24}$$

where all the variables are sampled as in the theorem (Equation 23). We build a CCA-valid adversary $\mathcal{B}$ with oracle access to $\mathcal{O} = (\mathbf{O}, \mathbf{u}, \mathbf{w})$ that on input $(1^n, pk, c)$ makes a list Que of at most $2^{n/4}$ queries and outputs b with probability at least $1/2 + \alpha(n)$. Note that $\mathcal{B}$ should never call $\langle \mathbf{u}, (pk, c)\rangle$. The proof of Theorem 9 then follows from Lemma 1.

We now show how to build $\mathcal{B}$ based on $\mathcal{A}$. The adversary $\mathcal{B}^{\mathcal{O}}(1^n, pk, c)$ invokes $\mathcal{A}^{\mathcal{O},\mathbf{T}}(1^n, pk, c)$, and reply to all $\mathcal{A}$'s $\mathcal{O}$ queries using its own $\mathcal{O}$ oracle. Note that since $\mathcal{A}$ is CCA valid it never makes the query $\langle \mathbf{u}, (pk, c)\rangle$.

To handle an $\mathcal{A}$'s query, $Tqu \stackrel{\text{def}}{=} \langle \mathbf{T}, (1^{n_1}, PK, C_1, \ldots, C_{n_1})\rangle$, $\mathcal{B}$ acts as follows:[8]

[8] Note that we are assuming that the query Tqu is made with respect to the security parameter 1^n. We made this assumption for clarity of presentation. Our proof below will not change if the query is made with respect to another security parameter.

1. **Sampling** $\mathsf{Freq} \leftarrow FreqQue^{\mathbf{O},\mathbf{u}}(PK, n^{23\vartheta})$: we show how $\mathcal{B}$ can do this. $\mathcal{B}^{\mathbf{O},\mathbf{u},\mathbf{w}}$ runs $E^{\mathbf{O}}(PK, 01)$ independently $n^{23\vartheta}$ times and add the symbolic versions all the query/response pairs to Freq. Moreover, for any $(\langle \mathfrak{d}, (sk', c')\rangle, *) \in \mathsf{Freq}$, $\mathcal{B}$ calls $\langle \mathbf{g}, sk'\rangle$ to get pk': if $pk = pk'$ then $\mathcal{B}$ has found its challenge secret key so it halts and returns $\langle \mathbf{d}, (sk', c)\rangle$. Otherwise, $\mathcal{B}$ calls $\langle \mathbf{u}, (pk', c')\rangle$: if the response is some $(b', r') \neq \perp$ then $\mathcal{B}$ adds $(\langle \mathfrak{e}, (pk, b', r')\rangle, c)$ to Freq. Finally, let $\mathsf{EmbedPut}$ contain any pk' such that $\langle \mathfrak{g}, *\rangle, pk') \in \mathsf{Freq}$ and assume without loss of generality that

$$pk \notin \mathsf{EmbedPub}. \tag{25}$$

   This is because otherwise $\mathcal{B}$ has found its challenge secret key and so it can halt and win the game.
2. $\mathcal{B}$ samples

$$(SK', \mathsf{Q_s}, \mathsf{Q_c}) \leftarrow ConsOrc(PK, \mathsf{Freq}), \tag{26}$$

   This step is done offline.
3. $\mathcal{B}$ starts executing $D^{\widetilde{\mathbf{O}}}(SK', C_1 \dots C_n)$, where

$$(\mathbf{e}_{imp}, \mathbf{d}_{imp}) = EncImpose(\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathsf{Q_c}),$$
$$(\widetilde{\mathbf{g}}, \widetilde{\mathbf{d}}) = KeyImpose(\mathbf{g}, \mathbf{e}_{imp}, \mathbf{d}_{imp}, \mathsf{Q_s})$$
$$\widetilde{\mathbf{e}} = \mathbf{e}_{imp} \text{ and } \widetilde{\mathbf{O}} = (\widetilde{\mathbf{g}}, \widetilde{\mathbf{e}}, \widetilde{\mathbf{d}}).$$

   We explain how $\mathcal{B}$ can handle this decryption by considering two possible sub-cases.
   - $(\langle \mathfrak{g}, *\rangle, pk) \notin \mathsf{Q_s}$: we claim that $\mathcal{B}$ can fully execute $D^{\widetilde{\mathbf{O}}}(SK', C_1, \dots, C_{n_1})$. First, by Part (a) of Lemma 7, $\mathcal{B}$ can efficiently handle all $(\widetilde{\mathbf{g}}, \widetilde{e})$ queries asked during the execution. For a query $qu = \langle \widetilde{\mathbf{d}}, (sk', c')\rangle$ asked, either (i) $(\langle \mathfrak{g}, sk'\rangle, *) \notin \mathsf{Q_s}$ or (ii) for some $pk' \neq pk$, $(\langle \mathfrak{g}, sk'\rangle, pk') \in \mathsf{Q_s}$. $\mathcal{B}$ can handle Case (i) by Part 1 of Proposition 1, and Case (ii) by Part 2 of Proposition 1. Note that for Case (ii) $\mathcal{B}$ needs to make a query $\langle \mathbf{u}, (pk', c')\rangle$, which is a valid query for $\mathcal{B}$ since $pk \neq pk'$. Now if $S_{out} = D^{\widetilde{\mathbf{O}}}(SK', C_1, \dots, C_{n_1})$, then $\mathcal{B}$ will perform Step 5 of $\mathbf{T}$, which $\mathcal{B}$ can successfully do since the rest only involves making $\mathbf{g}$ and $\mathbf{w}$ queries. Thus, $\mathcal{B}$ can find the answer to Tqu.
   - $(\langle \mathfrak{g}, *\rangle, pk) \in \mathsf{Q_s}$: In this case, recalling the definition of Qpub as given in $\mathbf{T}$'s description, we have $pk \in \mathsf{QPub}$, since $\mathbf{w}(pk) = \top$, which in turn is because pk is $\mathcal{B}$'s challenge public key and by definition is valid. Thus, by the condition given in Step 5 of $\mathbf{T}$'s description, if $S_{out} = D^{\widetilde{\mathbf{O}}}(SK', C_1, \dots, C_{n_1})$ then at least one of the following conditions holds:
     (a) The answer to Tqu is $\perp$; or
     (b) $pk \in \mathsf{EmbedPub}$; or
     (c) The query/response pair $(\langle \mathbf{g}, *\rangle, pk)$ will show up during $G^{\mathbf{O}}(S_{out})$.
     We already know by Equation 25 that $pk \notin \mathsf{EmbedPub}$. Thus, if

$$S_{out} = D^{\widetilde{\mathbf{O}}}(SK', C_1, \dots, C_{n_1})$$

     then one of the Conditions (a) and (c) above must hold.
     We now claim the following.

     Claim A. *Letting* $S_{out} = D^{\widetilde{\mathbf{O}}}(SK', C_1, \dots, C_n)$, $\mathcal{B}$ can find two strings $S_{out,0}$ and $S_{out,1}$ such that $S_{out} \in \{S_{out,0}, S_{out,1}\}$.

     If Claim A holds, $\mathcal{B}$ can execute both $G^{\mathbf{O}}(S_{out,0})$ and $G^{\mathbf{O}}(S_{out,1})$; if during either execution a query/response $(\langle \mathbf{g}, *\rangle, pk)$ is observed, $\mathcal{B}$ has learned $\mathbf{g}^{-1}(pk)$, winning the CCA game; otherwise, $\mathcal{B}$ in response to Tqu will return $\perp$, which is indeed the correct response. Thus, we are left to show why Claim A holds.

     Proof of Claim A. Assume

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

$\mathcal{B}$ forms the two sets

$$\mathsf{W} = \{(pk_1, c_1), \ldots, (pk_p, c_p)\}$$
$$\mathsf{W}' = \{(pk_1, \mathbf{e}(pk_1, b_1, r_1)), \ldots, (pk'_p, \mathbf{e}(pk_p, b_p, r_p))\},$$

which $\mathcal{B}$ can efficiently do. Now $\mathcal{B}$ will attempt to decrypt $D^{\widetilde{\mathbf{O}}}(SK', C_1, \ldots, C_{n_1})$. By Part (a) of Lemma 7 $\mathcal{B}$ can efficiently reply to both $\widetilde{\mathbf{g}}$ and $\widetilde{\mathbf{e}}$ queries made along the way. For a query $qu = \widetilde{\mathbf{d}}(sk', c')$, one of the following possibilities must hold:

- $(\langle \mathfrak{g}, sk' \rangle, *) \notin \mathsf{Q_s}$: in this case $\mathcal{B}$ can find the answer to qu by Part 1 of Proposition 1.
- $(\langle \mathfrak{g}, sk' \rangle, pk') \in \mathsf{Q_s}$ and $pk' \neq pk$: in this case $\mathcal{B}$ can find the answer to qu by Part 2 of Proposition 1.
- $(\langle \mathfrak{g}, sk' \rangle, pk) \in \mathsf{Q_s}$ and $c' \in \mathsf{W} \cup \mathsf{W}'$: in this case, by Parts (d) and (e) of Lemma 7, $\mathcal{B}$ can determine the answer to qu.
- $(\langle \mathfrak{g}, sk' \rangle, pk) \in \mathsf{Q_s}$, $c' \neq c$ and $c' \notin \mathsf{W} \cup \mathsf{W}'$: in this case by Part (c) of Lemma 7 the answer to qu can be determined by calling $\langle \mathbf{u}, (pk, c') \rangle$, which is a valid query for $\mathcal{B}$, as $c' \neq c$.
- $(\langle \mathfrak{g}, sk' \rangle, pk) \in \mathsf{Q_s}$, $c' = c$ and $c \notin \mathsf{W} \cup \mathsf{W}'$: in this case by Part (c) of Lemma 7 the answer to qu is the unknown challenge bit b. Thus, for the first time this case occurs, $\mathcal{B}$ will start two parallel branches BR_0 and BR_1 of computation, where $\mathcal{B}$ will reply to qu with b_1 on branch BR_{b_1}. On both branches $\mathcal{B}$ will reply to all $\widetilde{\mathbf{g}}$, $\widetilde{\mathbf{e}}$ and $\widetilde{\mathbf{d}}$ exactly as explained above. Moreover, if on some branch $BR_{b'}$, still during the execution of $D^{\widetilde{\mathbf{O}}}(SK', C_1, \ldots, C'_{n_1})$, the current query is asked again (i.e., the query $qu' = \langle \widetilde{\mathbf{d}}, (sk', c) \rangle$, where $(\langle \mathfrak{g}, sk' \rangle, pk) \in \mathsf{Q_s}$ and $c \notin \mathsf{W} \cup \mathsf{W}'$) $\mathcal{B}$ will reply to qu' with b', making it consistent with the previous reply.

Claim A now follows by the above way of simulation. □

At the end of $\mathcal{A}$'s simulation $\mathcal{B}$ outputs whatever $\mathcal{A}$ does. Note that $\mathcal{B}$ makes at most $2^{n/8} \times poly(n) \leq 2^{n/4}$ queries, since $\mathcal{B}$ replies to each $\mathcal{A}$'s query by making $poly(n)$ queries. Moreover, note that whenever $\mathcal{B}$ halts without completing the stimulation, it has found the challenge bit b. The proof of Theorem 9 is now complete. □

5.7 Putting all together

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

Theorem 10. *There exists no fully-blackbox construction of* 1*-seed circular-secure bit-encryption schemes from CPA-secure encryption schemes.*

To prove the above theorem we first give and prove the following lemma.

Lemma 8. *For any bit encryption construction (G, E, D), there exists a PPT oracle adversary $\mathcal{A}$, where*

$$\Pr_{\mathbf{O},\mathbf{T}}\left[\mathcal{A}^{\mathbf{O},\mathbf{T}} \text{ breaks the 1-seed circular security of } (G^{\mathbf{O}}, E^{\mathbf{O}}, D^{\mathbf{O}})\right] = 1 \tag{27}$$

The proof of the above lemma, given below, is based on a simple averaging argument and an easy application of the Borel-Cantelli Lemma. See [MMN+16] for more specialized variants of the Borel-Cantelli Lemma useful for separation theorems.

Proof. First, note that by Theorem 5 there exists a PPT oracle adversary $\mathcal{A}$ such that

$$\alpha(n) \stackrel{\text{def}}{=} \Pr_{\mathbf{O},\mathbf{T},S,(C_1,\ldots,C_n)}\left[\mathcal{A}^{\mathbf{O},\mathbf{T}}(1^n, PK, C_1, \ldots, C_n) = S\right] \geq 1 - \frac{1}{n^5},$$

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

Using a simple averaging argument we obtain

$$\beta(n) \stackrel{\text{def}}{=} \Pr_{\mathbf{O},\mathbf{T}}\left[\Pr_{S,(C_1,\ldots,C_n)}\left[\mathcal{A}^{\mathbf{O},\mathbf{T}}(1^n, PK, C_1, \ldots, C_n) = S\right] \geq 1 - \frac{1}{n^3}\right] \geq 1 - \frac{1}{n^2}. \tag{28}$$

To see why the above equation holds let fr denote the fraction of $(\mathbf{O},\mathbf{T})$ for which it holds

$$\Pr_{S,(C_1,\ldots,C_n)}\left[\mathcal{A}^{\mathbf{O},\mathbf{T}}(1^n, PK, C_1, \ldots, C_n) = S\right] < 1 - \frac{1}{n^3}.$$

We want to show that $fr \leq \frac{1}{n^2}$. Assume to the contrary that $fr > \frac{1}{n^2}$. Then

$$\alpha(n) < fr(1 - \frac{1}{n^3}) + (1 - fr) \times 1 = 1 - fr \times \frac{1}{n^3} < 1 - \frac{1}{n^5},$$

which is a contradiction.

Now Equation 28 implies that for at most a $\frac{1}{n^2}$ fraction of $(\mathbf{O},\mathbf{T})$ the adversary $\mathcal{A}^{\mathbf{O},\mathbf{T}}$, on security parameter 1^n, outputs S with probability less than $1 - \frac{1}{n^3}$. Since $\sum_{n=0}^{\infty} \frac{1}{n^2}$ converges, by the Borel-Cantelli Lemma we obtain that for measure-one of oracles $(\mathbf{O},\mathbf{T})$ the adversary $\mathcal{A}^{\mathbf{O},\mathbf{T}}$ breaks the 1-seed circular security of $(G^{\mathbf{O}}, E^{\mathbf{O}}, D^{\mathbf{O}})$: for all sufficiently large n the adversary $\mathcal{A}^{\mathbf{O},\mathbf{T}}$ recovers S from $(PK, C_1, \ldots, C_n)$ with probability at least $1 - \frac{1}{n^3}$. □

Proof (of Theorem 10). Suppose to the contrary that such a reduction exists and let (G, E, D) and *Red* be, respectively, the associated construction and the security-proof algorithms. By Lemma 8 there exists a PPT-oracle adversary $\mathcal{A}$ such that

$$\Pr_{\mathbf{O}=(\mathbf{g},\mathbf{e},\mathbf{d}),\mathbf{T}}\left[\mathcal{A}^{\mathbf{O},\mathbf{T}} \text{ breaks the 1-seed circular security of } (G^{\mathbf{O}}, E^{\mathbf{O}}, D^{\mathbf{O}})\right] = 1.$$

That is, there exists a PPT-oracle adversary $\mathcal{A}$ such that for a measure one of oracles $\mathcal{OR}$ we have that for any $(\mathbf{g},\mathbf{e},\mathbf{d},\mathbf{T}) \in \mathcal{OR}$, it holds that $\mathcal{A}^{\mathbf{O},\mathbf{T}}$ breaks the 1-seed circular security of $(G^{\mathbf{O}}, E^{\mathbf{O}}, D^{\mathbf{O}})$. Now, letting $\alpha(n) = n^{-\log n}$ (it can be any value so long as $\alpha(n) >> \frac{1}{2^{n/4}}$) we obtain that for any $(\mathbf{g},\mathbf{e},\mathbf{d},\mathbf{T}) \in \mathcal{OR}$ and for infinitely many n:

$$\Pr\left[Red^{(\mathcal{A}^{\mathbf{g},\mathbf{e},\mathbf{d},\mathbf{T}})}(1^n, pk, c) = b\right] \geq \frac{1}{2} + \alpha(n), \tag{29}$$

where $b \leftarrow \{0,1\}$, $sk \leftarrow \{0,1\}^n$, $pk = \mathbf{g}(sk)$ and $c \leftarrow \mathbf{e}(pk, b)$.

Now since both *Red* and $\mathcal{A}$ are PPT, there exists a PPT adversary $\mathcal{B}$ such that for any $(\mathbf{g},\mathbf{e},\mathbf{d},\mathbf{T}) \in \mathcal{OR}$ and for infinitely many n:

$$\Pr_{b,sk,c}\left[\mathcal{B}^{\mathbf{g},\mathbf{e},\mathbf{d},\mathbf{T}}(1^n, pk, c) = b\right] \geq \frac{1}{2} + \alpha(n). \tag{30}$$

Now since $\mathcal{OR}$ is a measure-one subset of the set of all oracles, the following holds for infinitely many n:

$$\Pr_{\mathbf{g},\mathbf{e},\mathbf{d},\mathbf{T},b,sk,c}\left[\mathcal{B}^{\mathbf{O},\mathbf{T}}(1^n, pk, c) = b\right] \geq \frac{1}{2} + \alpha(n),$$

where $(\mathbf{g},\mathbf{e},\mathbf{d},*,*) \leftarrow \Psi$ and other variables are sampled as above. However, this contradicts Theorem 9, and the proof is complete. □

5.8 Extensions of the separation result

Separating 1-weak-seed circular security from CCA2 security. We note that our septation proved in Theorem 10 extends even if the base bit-encryption scheme is CCA2 secure and the constructed scheme is only required to satisfy a weak form of 1-seed circular security, in which the adversary now should recover the underlying seed. We define this notion below.

Definition 9. *Following the notation of Definition 1 we call $\mathcal{E} = (G, E, D)$ t-weak-seed circular secure if for all PPT adversaries $\mathcal{A}$:*

$$\Pr[\mathcal{A}(\mathbf{InpSeed}) = S_1] = negl(n).$$

To see why our separation extends to the above setting, first note that the oracle $\mathbf{T}$, which is used in Lemma 8, returns the seed as output, helping us to indeed break 1-weak-seed circular security. Moreover, the proof of Theorem 9 indeed shows the stronger result in which $\mathcal{A}$ is allowed to make CCA queries.

We conclude the discussion about this extension with two remarks. Firstly, note that for CCA2 security in the setting of bit encryption, we may assume without loss of generality that the adversary will make all its CCA queries after receiving the challenge ciphertext. This is exactly how we model CCA adversaries in our proofs (e.g., proof of Theorem 9). Secondly, by the result of Myers and Shelat [MS09] we know that CCA2-secure many bit encryption can be based on CCA2-secure bit encryption, and so restricting our base CCA-secure primitive to be bit encryption is without loss of generality.

Beyond single-bit 1-seed circular secure encryption. 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.

We first sketch how to adjust $\mathbf{T}$ to make our separation work for the case in which (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 of its execution, $\mathbf{T}$ will call this new variant 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). The proofs of the other lemmas that lead up to Theorem 5, as well as the proof of Theorem 9, remain unchanged.

Finally, note that the above extension heavily relies on the fact that the message size is $O(\log n)$, and so the idea fails 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 Subsection 4.2. Throughout this section note that (G, E, D) has the same plaintext and seed space. Most of the tools underlying $\mathbf{T_{t+1}}$ have been presented before, but we will need the following extension of Definition 6.

Definition 10. *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 $(\langle \mathfrak{d}, (sk, c)\rangle, *) \in$ Freq if $\mathbf{u}(\mathbf{g}(sk), c) = (b, r) \neq \perp$ add $(\langle \mathfrak{e}, (pk, b, r)\rangle, c)$ to* Freq*.*

Remark 1. Throughout the remaining sections we will 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: $(1^n, PK_1, \ldots, PK_{t+1}, C_1, \ldots, C_{t+1})$

Operations:

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 such that $(\langle \mathfrak{g}, * \rangle, pk) \in \mathsf{Freq_i}$.
2. **Sampling consistent oracles/secret-keys:** For $i \leq t+1$ sample
$$(\widetilde{SK}_i, \mathsf{Q_s^i}, \mathsf{Q_c^i}) \leftarrow ConsOrc(PK_i, \mathsf{Freq_i}), \tag{31}$$
and let $\mathsf{QPub_i}$ contain any pk such that $\mathbf{w}(pk) = \top$ and $(\langle \mathfrak{g}, * \rangle, pk) \in \mathsf{Q_s^i}$.
3. If for some distinct $i, j \in [t+1]$:
$$(\mathsf{QPub_i} \cap \mathsf{QPub_j}) \setminus (\mathsf{FreqPub_i} \cup \mathsf{FreqPub_j}) \neq \emptyset,$$
then halt and return $\perp$.
4. **Defining intermediate oracles:** For $i \leq t+1$ define
$$(\mathbf{e}_{imp,i}, \mathbf{d}_{imp,i}) = EncImpose(\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathsf{Q_c^i})$$
$$(\widetilde{\mathbf{g}}_i, \widetilde{\mathbf{d}}_i) = KeyImpose(\mathbf{g}, \mathbf{e}_{imp,i}, \mathbf{d}_{imp,i}, \mathsf{Q_s^i}).$$
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 $\mathbf{T}_{\mathbf{t+1}}$.

Environment: $Env_{t+1}(n)$: Output
$$(\mathbf{O}, \mathbf{u}, \mathbf{w}, S_1, \ldots, S_{t+1}, PK_1, \ldots, PK_{t+1}, E^{\mathbf{O}}(PK_1, S_2), \ldots, E^{\mathbf{O}}(PK_{t+1}, S_1)),$$
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 $1 \leq i \leq t+1$.

6.1 $\mathbf{T_{t+1}}$ breaks the $(t+1)$-seed circular security of (G, E, D)

We show that with high probability $\mathbf{T_{t+1}}$ will return the desired output if used honestly.

Theorem 11. *It holds that*

$$\Pr_{\mathbf{Env},\mathbf{T_{t+1}}}[\mathbf{T_{t+1}}(1^n, PK_1, \ldots, PK_{t+1}, C_1, \ldots, C_{t+1}) = S_1] \geq 1 - \frac{1}{n^5}, \tag{32}$$

where

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

To help us describe the lemmas that lead up to the proof of Theorem 11, we define the following bad events that can occur during the execution of $\mathbf{T}_{t+1}$.

Definition 11. *For*

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

we define

- *Bad: the event that the execution of* $\mathbf{T_{t+1}}(1^n, PK_1, \ldots, PK_{t+1}, C_1, \ldots, C_{t+1})$ *fails due to Line 3: That is, for some distinct* $i, j \in [t+1]$

$$(\mathsf{QPub_i} \cap \mathsf{QPub_j}) \setminus (\mathsf{FreqPub_i} \cup \mathsf{FreqPub_j}) \neq \emptyset.$$

- *DecFail: the event that* $S_1 \neq \widetilde{D^{\mathbf{O}_{t+1}}}(SK'_{t+1}, C_{t+1})$ *or for some* $2 \leq i \leq t+1$, $S_i \neq \widetilde{D^{\mathbf{O}_{i-1}}}(SK'_{i-1}, C_{i-1})$.

We bound the probability of the above two undesirable events in the following lemma.

Lemma 9. *It holds that*

1. $\Pr_{\mathbf{Env},\mathbf{T_{t+1}}}[Bad] \leq \frac{1}{n^9}$,
2. $\Pr_{\mathbf{Env},\mathbf{T_{t+1}}}[DecFail] \leq \frac{1}{n^6}$,

where $\mathbf{Env} \leftarrow Env_{t+1}(n)$

The proof of Part 1 of Lemma 9 follows exactly along the same lines as the proof sketch given in Subsection 4.2. Also, the proof of Part 2 follows similarly to that of Lemma 3. We give the proof of the Lemma in Section D of the appendix.

We now bound the probability that $\mathbf{T_{t+1}}$, on an honest input, returns $\perp$ while $\overline{Bad} \wedge \overline{DecFail}$ holds.

Lemma 10.

$$\Pr_{\mathbf{Env},\mathbf{T_{t+1}}}\left[\overline{Bad} \wedge \overline{DecFail} \wedge (\mathbf{T_{t+1}}(1^n, PK_1, \ldots, PK_{t+1}, C_1, \ldots, C_{t+1}) = \perp)\right] \leq \frac{1}{2^n},$$

where

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

We first show how to use Lemmas 9 and 10 to prove Theorem 11. Then, we will prove Lemma 10.

Proof (of Theorem 11). All probabilities below are taken over the variables sampled in the theorem. Let

$$\mathbf{inp} = (1^n, PK_1, \ldots, PK_{t+1}, C_1, \ldots, C_{t+1}).$$

Let $Evnt$ be the event that $\mathbf{T_{t+1}}(\mathbf{inp}) \neq S_1$, which we want to bound. We claim if $Evnt$ holds then at least one of the events $Evnt2$ and $Evnt3$, defined below, must hold:

– $Evnt2 = Bad \vee DecFail$.
– $Evnt3 = \overline{Bad} \wedge \overline{DecFail} \wedge (\mathbf{T_{t+1}}(\mathbf{inp}) = \perp)$.

The proof of the above claim is via contraposition: if $\overline{Evnt2} \wedge \overline{Evnt3}$ holds, then :

(I) $S^1_{out} = S_1$. This is because $\overline{Bad} \wedge \overline{DecFail}$ holds.
(II) $\mathbf{T_{t+1}}(\mathbf{inp}) \neq \perp$.

By the way $\mathbf{T}$ is designed if (I) and (II) hold then we definitely have $\mathbf{T_{t+1}}(\mathbf{inp}) = S_1$.

Thus, $\Pr[Evnt] \leq \Pr[Evnt2 \vee Evnt3]$. By Lemmas 9 and 10 $\Pr[Evn2] \leq \frac{2}{n^6}$ and $\Pr[Evnt3] \leq \frac{1}{2^n}$. The claimed bound follows. □

Proof (of Lemma 10).
We need to show

$$\alpha(n) \stackrel{\text{def}}{=} \Pr_{\mathbf{Env},\mathbf{T_{t+1}}}\left[\overline{Bad} \wedge \overline{DecFail} \wedge (\mathbf{T_{t+1}}(1^n, PK_1, \ldots, PK_{t+1}, C_1, \ldots, C_{t+1}) = \perp)\right] \leq \frac{1}{2^n},$$

where

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

Let Nil be the event that

$$\mathbf{T_{t+1}}(1^n, PK_1, \ldots, PK_{t+1}, C_1, \ldots, C_{t+1}) = \perp.$$

We claim the following:

Claim I. If all of $\overline{Bad}$, $\overline{DecFail}$ and Nil hold, then the event $NSubSet$, defined below, holds:

$$NSubSet \stackrel{\text{def}}{=} \text{ for some } 1 \leq i \leq t+1 : \mathsf{QPub_i} \not\subseteq \mathsf{EmbedPub_i} \cup \mathsf{FreqPub_i}.$$

We first show how to use Claim I to prove Lemma 10 and then show why the claim holds. By Claim I we have

$$\alpha(n) \leq \beta(n) \stackrel{\text{def}}{=} \Pr[\text{for some } 1 \leq i \leq t : \mathsf{QPub_i} \not\subseteq \mathsf{EmbedPub_i} \cup \mathsf{FreqPub_i}]. \tag{33}$$

We can use exactly the same proof as that of Lemma 4 to show that for any fixed i:

$$\Pr[\mathsf{QPub_i} \not\subseteq \mathsf{EmbedPub_i} \cup \mathsf{FreqPub_i}] \leq \frac{1}{2^{2n}}. \tag{34}$$

Thus, we have

$$\beta(n) \leq \frac{n^{t_0}}{2^{2n}} \leq \frac{1}{2^n},$$

as desired.

We now prove Claim I. Suppose all of $\overline{Bad}$, $\overline{DecFail}$ and Nil hold. We show that this implies that $NSubSet$ must necessarily hold. Since the event Nil holds, then at least one of the following must hold:

(A) The bad event in Line 3 of of $\mathbf{T_{t+1}}$'s computation holds;
(B) $S^1_{out} = \perp$.
(C) The bad event in Line 6 of of $\mathbf{T_{t+1}}$'s computation holds;

Since we know $\overline{Bad}$ and $\overline{DecFail}$ hold, the events (A) and (B) above cannot hold. Thus, the event (C) must necessarily hold, which is exactly the event $NSubSet$. The claim now follows and the proof is complete. □

6.2 $\mathbf{T_{t+1}}$ does not break the t-seed circular security of $(\mathbf{g}, \mathbf{e}, \mathbf{d})$

The following theorem shows that $\mathbf{T_{t+1}}$ is not helpful in breaking the t-seed circular security of the base bit encryption scheme $(\mathbf{g}, \mathbf{e}, \mathbf{d})$. The proof generalizes a proof sketch given for the case $t = 1$ in Subsection 4.2.

Theorem 12. *Suppose $\mathcal{A}$ is a t-seed circular security adversary with access to the oracles $\mathbf{O} = (\mathbf{g}, \mathbf{e}, \mathbf{d})$ and $\mathbf{T_{t+1}}$ that makes at most $2^{n/8}$ queries. We have*

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

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 $i \leq t$.

Proof. The proof uses some of the ideas discussed extensively before (especially in the proof of Theorem 9) and so we omit the details whenever appropriate.

Let

$$\mathbf{inp} = (1^n, pk_1, \ldots, pk_t, \{(c_1^i, \ldots, c_n^i)\}_{i \in [t]}, c)$$

be $\mathcal{A}$'s input, sampled as specified in the theorem. That is , for $1 \leq i \leq t-1$:

$$(c_1^i, \ldots, c_n^i) \leftarrow \mathbf{e}_{pk_i}(sk_{i+1})$$

and

$$(c_1^t, \ldots, c_n^t) \leftarrow \mathbf{e}_{pk_t}(sk_1).$$

From $\mathcal{A}$ we show how to build an adversary $\mathcal{B}$ which has access to the oracles $(\mathbf{O}, \mathbf{u}, \mathbf{w})$ and which on input $\mathbf{inp}$ manages to recover b with at least the same advantage as $\mathcal{A}$ and which furthermore satisfies the following:

1. $\mathcal{B}$ is CCA-valid (Definition 4). Namely, $\mathcal{B}$ never calls $\langle \mathbf{u}, (pk', c') \rangle$ for
$$(pk', c') \in \{(pk_1, c_1^1), \ldots, (pk_1, c_n^1), \ldots, (pk_t, c_1^t), \ldots, (pk_t, c_n^t)\},$$
2. $\mathcal{B}$ makes at most $poly(n) \times 2^{n/8}$ queries. (Recall that $2^{n/8}$ is the maximum number of queries that $\mathcal{A}$ is allowed to make.)

By invoking Lemma 1 we will then obtain that the advantage of such an adversary $\mathcal{B}$ is at most $\frac{1}{2} + \frac{1}{2^{n/4}}$, obtaining the bound of the theorem.

To build $\mathcal{B}$ we show, as before, how to handle $\mathbf{T_{t+1}}$ queries of $\mathcal{A}$ using $\mathbf{u}$ and $\mathbf{w}$ queries. Let

$$Tqu = \langle \mathbf{T_{t+1}}, (1^n, PK_1, \ldots, PK_{t+1}, C_1, \ldots, C_{t+1}) \rangle$$

be a query of $\mathcal{A}$.[9] To reply to Tqu, $\mathcal{B}^{\mathbf{O}, \mathbf{u}, \mathbf{w}}(\mathbf{inp})$ acts as follows:

1. $\mathcal{B}^{\mathbf{O}, \mathbf{u}, \mathbf{w}}$ performs Steps 1 and 2 of $\mathbf{T_{t+1}}$'s computation, which $\mathcal{B}^{\mathbf{O}, \mathbf{u}, \mathbf{w}}$ can perfectly do by making $\mathbf{O}$ and $\mathbf{w}$ queries. Without loss of generality we may assume that for all $1 \leq i \leq t$
$$pk_i \notin \mathsf{FreqPub}_1 \cup \cdots \cup \mathsf{FreqPub}_{t+1}, \tag{35}$$
because otherwise $\mathcal{B}^{\mathbf{O}, \mathbf{u}, \mathbf{w}}$ has found the matching secret key of pk_i (which is one of its challenge public keys), winning the game.
2. $\mathcal{B}^{\mathbf{O}, \mathbf{u}, \mathbf{w}}$ performs Step 3 of $\mathbf{T_{t+1}}$'s computation: if the check in this step fails $\mathcal{B}$ stops and returns $\perp$. Thus, assume for all distinct $i, j \in \{1, \ldots, t+1\}$
$$(\mathsf{QPub}_i \cap \mathsf{QPub}_j) \setminus (\mathsf{FreqPub}_i \cup \mathsf{FreqPub}_j) = \emptyset. \tag{36}$$

[9] Here for simplicity we assume the underlying query to $\mathbf{T_{t+1}}$ is relative to the security parameter n. The proof remains unchanged if the query is with respect to another security parameter.

3. For Steps 4 and 5 we consider two cases:
   - For all $1 \leq i \leq t+1$
   $$\mathsf{Q}_{\mathsf{s}}^{\mathsf{i}} \cap \{pk_1, \ldots, pk_t\} = \emptyset :$$
   in this case $\mathcal{B}$ can perfectly execute all decryptions: Namely, $S_{out}^1 = D^{\widetilde{\mathbf{O}_{t+1}}}(\widetilde{SK_{t+1}}, C_{t+1})$ and $S_{out}^i = D^{\widetilde{\mathbf{O}_{i-1}}}(\widetilde{SK_{i-1}}, C_{i-1})$ for $2 \leq i \leq t+1$. (All $\widetilde{\mathbf{d}}_i$ queries can be handled by Proposition 1 and all $\widetilde{\mathbf{g}_i}$ and $\widetilde{\mathbf{e}_i}$ queries can be handled by Item (a) of Lemma 7.) Having retrieved all of $S_{out}^1, \ldots, S_{out}^{t+1}$, $\mathcal{B}^{\mathbf{O},\mathbf{u},\mathbf{w}}$ can perform the last step (i.e., Step 6) of $\mathbf{T_{t+1}}$'s computation, which $\mathcal{B}^{\mathbf{O},\mathbf{u},\mathbf{w}}$ can perfectly do as this step only involves making $\mathbf{O}$ queries and doing simple offline computations.
   - For some $1 \leq i \leq t+1$:
   $$\mathsf{Q}_{\mathsf{s}}^{\mathsf{i}} \cap \{pk_1, \ldots, pk_t\} \neq \emptyset.$$
   In this case we will show that the answer to Tqu is either $\perp$ or $\mathcal{B}^{\mathbf{O},\mathbf{u},\mathbf{w}}$ is able to find the corresponding secret key of one of its challenge public keys.
   To show this, we first claim there exists $1 \leq h \leq t+1$ such that
   $$\begin{aligned} &\mathsf{Q}_{\mathsf{s}}^{\mathsf{h}} \cap \{pk_1, \ldots, pk_t\} = \emptyset \text{ and} \\ &\mathsf{Q}_{\mathsf{s}}^{\mathsf{h}'} \cap \{pk_1, \ldots, pk_t\} \neq \emptyset, \end{aligned} \tag{37}$$
   where $h' = 1$ if $h = t+1$ and $h' = h+1$, otherwise.
   The reason behind the above claim is that if for all $i \leq t+1$: $\mathsf{Q}_{\mathsf{s}}^{\mathsf{i}} \cap \{pk_1, \ldots, pk_t\} \neq \emptyset$, by the Pigeonhole Principle there is $pk \in \{pk_1, \ldots, pk_t\}$ and distinct indices $i, j \in [t+1]$ such that $pk \in \mathsf{Q}_{\mathsf{s}}^{\mathsf{i}} \cap \mathsf{Q}_{\mathsf{s}}^{\mathsf{j}}$. On the other hand, by Equation 35 $pk \notin \mathsf{FreqPub_i} \cup \mathsf{FreqPub_j}$. Thus, Equation 36 is violated.

   Now equipped with Equation 37 $\mathcal{B}^{\mathbf{O},\mathbf{u},\mathbf{w}}$ acts as follows:
   - $\mathcal{B}^{\mathbf{O},\mathbf{u},\mathbf{w}}$ executes $D^{\widetilde{\mathbf{O}_h}}(\widetilde{SK_h}, C_h)$ to obtain S'. Note that $\mathcal{B}^{\mathbf{O},\mathbf{u},\mathbf{w}}$ can successfully perform this execution. Again, this follows because all $\widetilde{\mathbf{d}_h}$ queries can be handled by Proposition 1 and all $\widetilde{\mathbf{g}_h}$ and $\widetilde{\mathbf{e}_h}$ queries can be handled by Item (a) of Lemma 7.)
   - $\mathcal{B}^{\mathbf{O},\mathbf{u},\mathbf{w}}$ executes $G^{\mathbf{O}}(S')$: if during the execution a query/response pair $(\langle \mathbf{g}, * \rangle, pk)$ for $pk \in \{pk_1, \ldots, pk_t\}$ turns up, then $\mathcal{B}^{\mathbf{O},\mathbf{u},\mathbf{w}}$ has found the secret key of one of its challenge public keys, winning the game. If no such a query/response pair turns up $\mathcal{B}^{\mathbf{O},\mathbf{u},\mathbf{w}}$ halts and returns $\perp$. By condition specified in Line 6 of $\mathbf{T_{t+1}}$'s execution, if the answer to Tqu is not $\perp$, then a query/response pair $(\langle \mathbf{g}, * \rangle, pk)$, for some $pk \in \{pk_1, \ldots, pk_t\}$, must occur during the execution of $G^{\mathbf{O}}(S')$, since $\mathsf{Q}_{\mathsf{s}}^{\mathsf{h}'} \cap \{pk_1, \ldots, pk_t\} \neq \emptyset$. Again recall that for all $1 \leq i \leq t$:
   $$pk_i \notin \mathsf{FreqPub_1} \cup \cdots \cup \mathsf{FreqPub_{t+1}}, \tag{38}$$

Thus, $\mathcal{B}^{\mathbf{O},\mathbf{u},\mathbf{w}}$, by being able to fully simulate $\mathcal{A}$ or else finding a corresponding secret key of one of it challenge public keys along the way, achieves at least the same advantage as $\mathcal{A}$, and the proof is complete. □

6.3 Putting all together

The following theorem is now established.

Theorem 13. *There exists no fully-blackbox construction of full-length $(t+1)$-seed circular security from bit t-seed circular security.*

The proof of the above theorem is obtained by combining Theorems 11 and 12 in exactly the same way the proof of Theorem 10 was established.

7 Partial separation of circular and CPA security

We show how our result, separating 1-seed circular secure bit encryption from CPA-secure encryption, extends to rule out a class of constructions for circular secure bit encryption, which we call *key-isolating constructions*. To define this class we first define it in a related model that we call the *canonical model*, and then we define this class 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) such that 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 separation between 1-circular secure bit encryption and CPA-secure encryption 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. Put differently, the last statement says that if there indeed exists a positive construction in the standard model, one can also be given in the CF model.

CF Key-isolating constructions. We call a CF construction (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 result extends to rule out all key-isolating constructions of circular-secure bit encryption from CPA-secure encryption in the CF model. The main idea behind this is that in a CF construction $(GS^{\mathbf{gs},\mathbf{e},\mathbf{d}}, GP^{\mathbf{O}}, E^{\mathbf{O}}, D^{\mathbf{O}})$, all the underlying $\mathbf{gp}$ queries made during the production of a pair of secret/public keys (SK, PK) are made during the execution of $GP^{\mathbf{O}}(SK)$. In other words, knowledge of SK enables us to reproduce all pk that are "embedded" in PK.

To adapt our results to rule out key-isolating circular-secure constructions, 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 CPA 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 straightforward 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 $(\mathbf{gs}, \mathbf{gp}, \mathbf{e}, \mathbf{d})$, as opposed to a standard PKE $(\mathbf{g}, \mathbf{e}, \mathbf{d})$. Again, as explained above this is without loss of generality.

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 such that:

- (GS, GP, E, D) is key-isolating; and
- (GS, GP) induces the same distribution as G. That is, for any $\mathbf{O} = (\mathbf{gs}, \mathbf{gp}, \mathbf{e}, \mathbf{d})$ the pair (SK, PK) is identically distributed to (SK', PK'), where

$$(SK, PK) \leftarrow G^{\mathbf{O}}(1^n), \quad SK' \leftarrow GS^{\mathbf{gs},\mathbf{e},\mathbf{d}}(1^n), \quad PK' = GP^{\mathbf{O}}(SK').$$

Now the impossibility of CF key-isolating constructions extends to standard ones, by how the counterpart notion is defined.

Examples. Any standard construction $\mathcal{E} = (G, E, D)$ under which seeds and secret keys are the same is key-isolating. To see this, define

$$GS(S) = S \quad GP^{\mathbf{gs},\mathbf{gp},\mathbf{e},\mathbf{d}}(S) = G_2^{\mathbf{gs},\mathbf{gp},\mathbf{e},\mathbf{d}}(S)$$

where G_2 is the algorithm that corresponds to the public-key output of G. We can now easily see that the construction $\mathcal{E}_{cf} = (GS, GP, E, D)$ is a CF-counterpart of (G, E, D), and that $\mathcal{E}_{cf}$ is key-isolating since GS makes no oracle calls at all. This shows that $\mathcal{E}$ is a key-isolating construction.

Using the same argument we can show that *identical constructions* are also key isolating, where these constructions are of the form $(G^{\mathbf{gs},\mathbf{gp}}, E^{\mathbf{e}}, D^{\mathbf{d}})$, where

- $G^{\mathbf{gs},\mathbf{gp}}(s) = (\mathbf{gs}(s), \mathbf{gp}(\mathbf{gs}(s)))$;
- $E^{\mathbf{e}}(pk, b) = \mathbf{e}(pk, b)$ and $D^{\mathbf{d}}(sk, c) = \mathbf{d}(sk, c)$.

Thus, we can explain the blackbox septation result of [Rot13] as a special case of our results.

8 Discussion related to the 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 with respect to the simple type of constructions discussed in Section 4.1. In what follows all mentions of the oracle $\mathbf{T}$ for the 1-seed circular-security case refers to the oracle $\mathbf{T}$ defined in Section 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 1-seed circular-security case by providing a CCA-valid 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 $\mathbf{T}(1^n, PK, C_1, \ldots, C_n)$ that is not simulatable by a CCA-valid 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 Section 4, until $\mathbf{T}$ obtains $SK_{out} = D^{\widetilde{\mathbf{d}}}(SK', C_1 \ldots C_n)$, 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 $\mathbf{g}^{-1}(\mathsf{QPub})$ is "embedded" in SK_{out}, before releasing SK_{out}. (Recall the definition of QPub from $\mathbf{T}$'s definition in Section 4.) But this "embedding condition" seems hard to check. This check was easy for the 1-seed circular-security case since we could 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 $D^{\mathbf{d}}(SK_{out}, \cdot)$ 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 SK_{out} if $\mathsf{QPub} \subseteq \mathbf{g}(\mathsf{EmbedSec})$; 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 $\mathbf{T}(1^n, PK, C_1 \ldots C_n)$ 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.

9 Open problems

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

While our impossibility results pertain to the public-key setting we believe that private-key seed circular secure encryption can also be separated from the same public-key assumptions.

Acknowledgements. We would like to thank Mohammad Mahmoody for helpful conversations in an early stage of our work. We are also grateful to the anonymous reviewers for their comments that considerably improved the presentation of our paper.

References

ACPS09. B. Applebaum, D. Cash, C. Peikert, and A. Sahai. *Fast cryptographic primitives and circular-secure encryption based on hard learning problems.* In S. Halevi (ed.), *CRYPTO 2009*, vol. 5677 of *LNCS*. Springer, 2009.

AP16. N. Alamati and C. Peikert. *Three's compromised too: Circular insecurity for any cycle length from (Ring-)LWE.* In Robshaw and Katz [RK16].

App14. B. Applebaum. *Key-dependent message security: Generic amplification and completeness. J. Cryptol.*, 27(3):429, 2014.

AS. G. Asharov and G. Segev. *Limits on the power of indistinguishability obfuscation and functional encryption.* In V. Guruswami (ed.), *FOCS 2015*. IEEE Computer Society.

AS16. G. Asharov and G. Segev. *On constructing one-way permutations from indistinguishability obfuscation.* In *Theory of Cryptography*. Springer, 2016.

BBF13. P. Baecher, C. Brzuska, and M. Fischlin. *Notions of black-box reductions, revisited.* In K. Sako and P. Sarkar (eds.), *ASIACRYPT 2013, (I)*, vol. 8269 of *LNCS*. Springer, 2013.

BG10. Z. Brakerski and S. Goldwasser. *Circular and leakage resilient public-key encryption under subgroup indistinguishability.* In T. Rabin (ed.), *CRYPTO 2010*, vol. 6223 of *LNCS*. Springer, 2010.

BGI+12. B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang. *On the (im) possibility of obfuscating programs. Journal of the ACM (JACM)*, 59(2):6, 2012.

BGV14. Z. Brakerski, C. Gentry, and V. Vaikuntanathan. *(leveled) fully homomorphic encryption without bootstrapping. ACM Transactions on Computation Theory (TOCT)*, 6(3):13, 2014.

BHHO08. D. Boneh, S. Halevi, M. Hamburg, and R. Ostrovsky. *Circular-secure encryption from decision Diffie-Hellman.* In D. Wagner (ed.), *CRYPTO 2008*, vol. 5157 of *LNCS*. Springer, 2008.

BHW15. A. Bishop, S. Hohenberger, and B. Waters. *New circular security counterexamples from decision linear and learning with errors.* In T. Iwata and J. H. Cheon (eds.), *ASIACRYPT 2015*, vol. 9453 of *LNCS*. Springer, 2015.

BKSY11. Z. Brakerski, J. Katz, G. Segev, and A. Yerukhimovich. *Limits on the power of zero-knowledge proofs in cryptographic constructions.* In Y. Ishai (ed.), *TCC 2011*, vol. 6597 of *LNCS*. Springer, 2011.

BPR+08. D. Boneh, P. A. Papakonstantinou, C. Rackoff, Y. Vahlis, and B. Waters. *On the impossibility of basing identity based encryption on trapdoor permutations.* In *FOCS 2008*. IEEE Computer Society, 2008.

BSW11. D. Boneh, A. Sahai, and B. Waters. *Functional encryption: Definitions and challenges.* In *Theory of Cryptography Conference*. Springer, 2011.

BV11. Z. Brakerski and V. Vaikuntanathan. *Fully homomorphic encryption from ring-lwe and security for key dependent messages.* In *CRYPTO 2011*. 2011.

BV14. Z. Brakerski and V. Vaikuntanathan. *Efficient fully homomorphic encryption from (standard) lwe. SIAM Journal on Computing*, 43(2):831, 2014.

CCPY16. C. Cheng, K. Chung, G. Persiano, and B. Yang (eds.). *PKC 2016, (II)*, vol. 9615 of *LNCS*. Springer, 2016.

CDSMW08. S. G. Choi, D. Dachman-Soled, T. Malkin, and H. Wee. *Black-box construction of a non-malleable encryption scheme from any semantically secure one.* In *Theory of Cryptography*. Springer, 2008.

CGH12. D. Cash, M. Green, and S. Hohenberger. *New definitions and separations for circular security.* In M. Fischlin, J. A. Buchmann, and M. Manulis (eds.), *PKC 2012*, vol. 7293 of *LNCS*. Springer, 2012.

CLTV15. R. Canetti, H. Lin, S. Tessaro, and V. Vaikuntanathan. *Obfuscation of probabilistic circuits and applications*. In Y. Dodis and J. B. Nielsen (eds.), *TCC 2015*, vol. 9015 of *LNCS*. Springer, 2015.

DDN91. D. Dolev, C. Dwork, and M. Naor. *Non-malleable cryptography (extended abstract)*. In *STOC 1991*. 1991.

FLS90. U. Feige, D. Lapidot, and A. Shamir. *Multiple non-interactive zero knowledge proofs based on a single random string*. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, 1990.

Gen09. C. Gentry. *Fully homomorphic encryption using ideal lattices*. In M. Mitzenmacher (ed.), *STOC 2009*. ACM, 2009.

GGH+16. S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. *Candidate indistinguishability obfuscation and functional encryption for all circuits*. *SIAM Journal on Computing*, 45(3):882, 2016.

GGSW13. S. Garg, C. Gentry, A. Sahai, and B. Waters. *Witness encryption and its applications*. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. ACM, 2013.

GKM+00. Y. Gertner, S. Kannan, T. Malkin, O. Reingold, and M. Viswanathan. *The relationship between public key encryption and oblivious transfer*. In *FOCS 2000*. IEEE Computer Society, 2000.

GMM07. Y. Gertner, T. Malkin, and S. Myers. *Towards a separation of semantic and CCA security for public key encryption*. In S. P. Vadhan (ed.), *TCC 2007*, vol. 4392 of *LNCS*. Springer, 2007.

GMR01. Y. Gertner, T. Malkin, and O. Reingold. *On the impossibility of basing trapdoor functions on trapdoor predicates*. In *FOCS 2001*. IEEE Computer Society, 2001.

GMW87. O. Goldreich, S. Micali, and A. Wigderson. *How to play any mental game*. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*. ACM, 1987.

GMW91. O. Goldreich, S. Micali, and A. Wigderson. *Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems*. *Journal of the ACM (JACM)*, 38(3):690, 1991.

Gol11. O. Goldreich. *Basing non-interactive zero-knowledge on (enhanced) trapdoor permutations: The state of the art*. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*. Springer, 2011.

GR13. O. Goldreich and R. D. Rothblum. *Enhancements of trapdoor permutations*. *Journal of cryptology*, 26(3):484, 2013.

GSW13. C. Gentry, A. Sahai, and B. Waters. *Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based*. In *Advances in Cryptology–CRYPTO 2013*. Springer, 2013.

GW11. C. Gentry and D. Wichs. *Separating succinct non-interactive arguments from all falsifiable assumptions*. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*. ACM, 2011.

HH09. I. Haitner and T. Holenstein. *On the (im)possibility of key dependent encryption*. In O. Reingold (ed.), *TCC 2019*, vol. 5444 of *LNCS*. Springer, 2009.

HK15. M. Hajiabadi and B. Kapron. *Reproducible circularly-secure bit encryption: Applications and realizations*. In R. Gennaro and M. Robshaw (eds.), *CRYPTO 2015, (I)*, vol. 9215 of *LNCS*. Springer, 2015.

HKS16. M. Hajiabadi, B. Kapron, and V. Srinivasan. *On generic constructions of circularly-secure, leakage-resilient public-key encryption schemes*. In Cheng et al. [CCPY16].

HR04. C.-Y. Hsiao and L. Reyzin. *Finding collisions on a public road, or do secure hash functions need secret coins?* In M. K. Franklin (ed.), *CRYPTO 2004*, vol. 3152 of *LNCS*. Springer, 2004.

IKLP06. Y. Ishai, E. Kushilevitz, Y. Lindell, and E. Petrank. *Black-box constructions for secure computation*. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*. ACM, 2006.

IR89. R. Impagliazzo and S. Rudich. *Limits on the provable consequences of one-way permutations*. In D. S. Johnson (ed.), *STOC 1989*. ACM, 1989.

KW16. V. Koppula and B. Waters. *Circular security separations for arbitrary length cycles from LWE*. In Robshaw and Katz [RK16].

MM16. M. Mahmoody and A. Mohammed. *On the power of hierarchical identity-based encryption*. In M. Fischlin and J. Coron (eds.), *EUROCRYPT 2016 (II)*, vol. 9666 of *LNCS*. Springer, 2016.

MMN+16. M. Mahmoody, A. Mohammed, S. Nematihaji, R. Pass, and A. Shelat. *A note on black-box complexity of indistinguishability obfuscation*. *IACR Cryptology ePrint Archive*, 2016:316, 2016.

MP12. M. Mahmoody and R. Pass. *The curious case of non-interactive commitmentss*. In R. Safavi-Naini and R. Canetti (eds.), *CRYPTO 2012*, vol. 7417 of *LNCS*. Springer, 2012.

MPS16. A. Marcedone, R. Pass, and A. Shelat. *Bounded KDM security from iO and OWF*. In *SCN 2016*, vol. 9841 of *LNCS*. Springer, 2016.

MR95. R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, New York, NY, USA, 1995.

MS09. S. Myers and A. Shelat. *Bit encryption is complete.* In *Foundations of Computer Science, 2009. FOCS'09.* IEEE, 2009.
MTY11. T. Malkin, I. Teranishi, and M. Yung. *Efficient circuit-size independent public key encryption with KDM security.* In K. G. Paterson (ed.), *EUROCRYPT 2011*, vol. 6632 of *LNCS*. Springer, 2011.
NY90. M. Naor and M. Yung. *Public-key cryptosystems provably secure against chosen ciphertext attacks.* In *Proceedings of the twenty-second annual ACM symposium on Theory of computing.* ACM, 1990.
Pas11. R. Pass. *Limits of provable security from standard assumptions.* In *Proceedings of the forty-third annual ACM symposium on Theory of computing.* ACM, 2011.
Pas13. R. Pass. *Unprovable security of perfect NIZK and non-interactive non-malleable commitments.* In *Theory of Cryptography.* Springer, 2013.
PSV06. R. Pass, A. Shelat, and V. Vaikuntanathan. *Construction of a non-malleable encryption scheme from any semantically secure one. Advances in Cryptology-CRYPTO 2006*, 2006.
RK16. M. Robshaw and J. Katz (eds.). *CRYPTO 2016 (II)*, vol. 9815 of *LNCS*. Springer, 2016.
Rot13. R. D. Rothblum. *On the circular security of bit-encryption.* In A. Sahai (ed.), *TCC 2013*, vol. 7785 of *LNCS*. Springer, 2013.
RS10. A. Rosen and G. Segev. *Chosen-ciphertext security via correlated products. SIAM Journal on Computing*, 39(7):3058, 2010.
RTV04. O. Reingold, L. Trevisan, and S. Vadhan. *Notions of reducibility between cryptographic primitives.* In M. Naor (ed.), *TCC 2004*, vol. 2951 of *LNCS*. Springer, 2004.
Sim98. D. R. Simon. *Finding collisions on a one-way street: Can secure hash functions be based on general assumptions?* In *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding*, vol. 1403 of *Lecture Notes in Computer Science*. Springer, 1998.
SW14. A. Sahai and B. Waters. *How to use indistinguishability obfuscation: deniable encryption, and more.* In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing.* ACM, 2014.
Vah10. Y. Vahlis. *Two is a crowd? a black-box separation of one-wayness and security under correlated inputs.* In D. Micciancio (ed.), *TCC 2010*, vol. 5978 of *LNCS*. Springer, 2010.
VDGHV10. M. Van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan. *Fully homomorphic encryption over the integers.* In H. Gilbert (ed.), *EUROCRYPT 2010*, vol. 6110 of *LNCS*. Springer, 2010.
Wee16. H. Wee. *KDM-security via homomorphic smooth projective hashing.* In Cheng et al. [CCPY16].

A Omitted Proofs from Section 4

Proof (of Lemma 1). The proof of the lemma is based on the idea that the functions $\mathbf{g}$ and $\mathbf{e}$ are length increasing, and are chosen uniformly at random. The proof for the general case involves defining many random variables and events. Thus, we sketch why a simple special case of the lemma holds and the proof for the general case follows the same line of arguments.

We show

$$\Pr\left[\mathcal{B}^{\mathcal{O}}(1^n, pk, c) = b\right] \leq \frac{1}{2} + \frac{1}{2^{n/4}},$$

where $\mathcal{O} \leftarrow \Psi$, $b \leftarrow \{0,1\}$, $sk \leftarrow \{0,1\}^n$, $pk = \mathbf{g}(sk)$ and $c \leftarrow \mathbf{e}(pk, b)$. Let b_{out} be $\mathcal{B}(1^n, pk, c)$'s output and let Que be the set of $\mathcal{B}$'s query/response pairs at the end of the computation. For bit b' let $\mathsf{Known}_{b'}$ contain any c' such that $(\langle \mathbf{e}, (pk, b', *)\rangle, c') \in \mathsf{Que}$ or $(\langle \mathbf{u}, (pk, c')\rangle, \top) \in \mathsf{Que}$. In words, the set $\mathsf{Known}_{b'}$ contains all ciphertexts c' that $\mathcal{B}$ knows are encryptions of b' under its challenge public key pk. Without loss of generality assume that $\mathcal{B}$ makes exactly $2 \times 2^{n/4}$ queries but at the end $|\mathsf{Known}_0| = |\mathsf{Known}_1|$. (If $\mathcal{B}$ needs to, say, increase the size of Known_0 it can make queries of the form $\langle \mathbf{e}, (pk, 0, *)\rangle$ as many as needed.) Also, recall by Assumption 3 that any $\langle \mathbf{d}, (sk', *)\rangle$ query of $\mathcal{B}$ is preceded by $\langle \mathbf{g}, sk'\rangle$.

Let *pubhit* be the event that $(\langle \mathbf{g}, *\rangle, pk) \in \mathsf{Que}$ and let *ciphhit* be the event that $c \in \mathsf{Known}_0 \cup \mathsf{Known}_1$. By the facts that $\mathbf{g}_n \colon \{0,1\}^n \to \{0,1\}^{5n}$ and $\mathbf{e}_n \colon \{0,1\}^{5n} \times \{0,1\} \times \{0,1\}^n \to \{0,1\}^{7n}$, we can upperbound the probability of *pubhit* $\vee$ *ciphhit* as

$$\frac{2^{n/4+1}}{2^{5n}} + \frac{2^{n/4+1}}{2^n} \leq \frac{1}{2^{n/4+1}} + \frac{1}{2^{n/4+1}} = \frac{1}{2^{n/4}}. \tag{39}$$

Moreover, recalling that for Que it holds that $|\mathsf{Known}_0| = |\mathsf{Known}_1|$, we have

$$\Pr[b_{out} = b \mid \overline{pubhit} \wedge \overline{ciphhit}] = 1/2. \tag{40}$$

By combining Equations 39 and 40 the proof follows. □

Proof (of Lemma 2). The proof of the lemma uses simple probabilistic arguments so we omit most of the details. Assume $\mathcal{B}$ at the end of its computation calls $\mathbf{w}$ on the public key it outputs. This only increases the number of queries by one. At any point during $\mathcal{B}$'s computation with current query list Que we say $\mathcal{B}$'s next query produces a *hit* if the next query is of the form $\langle \mathbf{w}, pk \rangle$ and it holds that $pk \in \{0,1\}^{5n}$, $\mathbf{w}(pk) = \top$ and $(\langle \mathbf{g}, * \rangle, pk) \notin \mathsf{Que}$. The adversary $\mathcal{B}$ wins if during $\mathcal{B}$'s computation at least one hit occurs. We now bound the event that the ith query for some fixed i produces hit, provided there were no previous hits. Assume the ith query is $\langle \mathbf{w}, pk \rangle$, for $pk \in \{0,1\}^{5n}$. The probability that the ith query produces a hit given there were no previous hits is at most

$$\frac{2^n}{2^{5n} - 2^n} \leq \frac{1}{2^{3n+\frac{n}{2}}}.$$

In above we used the fact that the number of valid public keys of length $5n$ is 2^n and the number of all public keys of length n is 2^{5n}. Using a union bound, we may bound the probability of the theorem as

$$\frac{2^n + 1}{2^{3n+\frac{n}{2}}} \leq \frac{2^{n+1}}{2^{3n+\frac{n}{2}}} \leq \frac{1}{2^{2n}}.$$

□

B Proof of Lemma 5, Part (B)

Before giving the proof of Part (B) of Lemma 5 we need to give a lemma which characterizes when the response to a query (sk, c) differs between $\mathbf{d}$ and $\widetilde{\mathbf{d}}$.

Lemma 11. *Let*

$$(\mathbf{O}, \mathbf{u}, \mathbf{w}, PK) \in Env(n) \text{ and } (\mathsf{Freq}, SK', \mathsf{Q_s}, \mathsf{Q_c}, \widetilde{\mathbf{O}}) \in Tvars^{\mathcal{O}}(PK).$$

For (sk, c) suppose $\widetilde{\mathbf{g}}(sk) = \mathbf{g}(sk) = pk$, but $\mathbf{d}(sk, c) \neq \widetilde{\mathbf{d}}(sk, c)$. Then all the following conditions must hold:

1. $(\langle \mathfrak{e}, (pk, *, *) \rangle, c) \in \mathsf{Q_c}$;
2. $(\langle \mathfrak{e}, (pk, *, *) \rangle, c) \notin \mathsf{Freq}$; *and*
3. $(\langle \mathfrak{d}, (sk, c) \rangle, *) \notin \mathsf{Freq}$.

Proof. Suppose $\widetilde{\mathbf{g}}(sk) = \mathbf{g}(sk) = pk$ and $\mathbf{d}(sk, c) \neq \widetilde{\mathbf{d}}(sk, c)$. Recall that

$$\begin{aligned}
&(\mathbf{e}_{imp}, \mathbf{d}_{imp}) = EncImpose(\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathsf{Q_c}) \\
&(\widetilde{\mathbf{g}}, \widetilde{\mathbf{d}}) = KeyImpose(\mathbf{g}, \mathbf{e}_{imp}, \mathbf{d}_{imp}, \mathsf{Q_s}) \\
&\widetilde{\mathbf{e}} = \mathbf{e}_{imp}.
\end{aligned}$$

To prove the lemma we first prove the following claim.

Claim 14. *Assuming $\widetilde{\mathbf{g}}(sk) = \mathbf{g}(sk) = pk$ it holds that $\widetilde{\mathbf{d}}(sk, c) = \mathbf{d}_{imp}(sk, c)$*

Proof of Claim 14. Recall by Claim 7 that $(\mathbf{g}, \mathbf{e}_{imp}, \mathbf{d}_{imp})$ is a valid PKE. To show $\widetilde{\mathbf{d}}(sk, c) = \mathbf{d}_{imp}(sk, c)$ we consider two cases: $\widetilde{\mathbf{d}}(sk, c) \neq \perp$ and $\widetilde{\mathbf{d}}(sk, c) = \perp$.

We first consider the first case. For $b \in \{0, 1\}$ we have

$$\widetilde{\mathbf{d}}(sk, c) = b \Leftrightarrow \mathbf{e}_{imp}(pk, b, *) = c \Leftrightarrow \mathbf{d}_{imp}(sk, c) = b.$$

For the second case, we have

$$\widetilde{\mathbf{d}}(sk,c) = \bot \Leftrightarrow \text{for no } b\colon \mathbf{e}_{imp}(pk,b,*) = c \Leftrightarrow \mathbf{d}_{imp}(sk,c) = \bot.$$

Thus, we have $\widetilde{\mathbf{d}}(sk,c) = \mathbf{d}_{imp}(sk,c)$. □

We now show that the negation of any of the conditions claimed in Lemma 11 implies $\mathbf{d}(sk,c) = \mathbf{d}_{imp}(sk,c)$, which in turn by Claim 14 implies $\mathbf{d}(sk,c) = \widetilde{\mathbf{d}}(sk,c)$, which is a contradiction to the assumption of the lemma that $\mathbf{d}(sk,c) \neq \widetilde{\mathbf{d}}(sk,c)$.
¬(1) **holds:** Then immediately from the definition of $\mathbf{d}_{imp}$ we obtain that $\mathbf{d}(sk,c) = \mathbf{d}_{imp}(sk,c)$.
¬(2) **holds:** Then we have $(\langle \mathfrak{e}, (pk,b,*)\rangle, c) \in \mathsf{Freq}$, for some b. Thus, $\mathbf{d}(sk,c) = b$. Now in order for $\mathbf{d}(sk,c) \neq \mathbf{d}_{imp}(sk,c)$ to be true, it must hold that $(\langle \mathfrak{e}, (pk,1-b,*)\rangle, c) \in \mathsf{Q_c}$, which is impossible since $\mathsf{Q_c}$ agrees with Freq.
¬(3) **holds:** Thus, $(\langle \mathfrak{d}, (sk,c)\rangle, *) \in \mathsf{Freq}$. This in turn implies that either

- $(\langle \mathfrak{d}, (sk,c)\rangle, \bot) \in \mathsf{Freq}$, or
- $(\langle \mathfrak{d}, (sk,c)\rangle, b) \in \mathsf{Freq}$ for some bit b.

We consider both cases:

- If $(\langle \mathfrak{d}, (sk,c)\rangle, \bot) \in \mathsf{Freq}$, then by Assumption 3 we have $(\langle \mathfrak{g}, sk\rangle, pk) \in \mathsf{Freq}$. Now since $\mathsf{Q_c}$ is obtained from some $\mathbf{e}'$, where $(\mathbf{g}', \mathbf{e}', \mathbf{d}')$ is a valid PKE that agrees with Freq (see Definition 7) we have $(\langle \mathfrak{e}, (pk,*,*)\rangle, c) \notin \mathsf{Q_c}$. (To see this note that $\mathbf{d}'(sk,c) = \bot$ and that $\mathbf{g}'(sk) = pk$, since $(\langle \mathfrak{d}, (sk,c)\rangle, \bot) \in \mathsf{Freq}$ and $(\langle \mathfrak{g}, sk\rangle, pk) \in \mathsf{Freq}$. If $(\langle \mathfrak{e}, (pk,*,*)\rangle, c) \in \mathsf{Q_c}$ then $\mathbf{e}'(pk,*,*) = c$, and since $\mathbf{g}'(sk) = pk$, it holds that $\mathbf{d}'(sk,c) \neq \bot$, which is a contradiction to the earlier established fact that $\mathbf{d}'(sk,c) = \bot$.) Now the fact that $(\langle \mathfrak{e}, (pk,*,*)\rangle, c) \notin \mathsf{Q_c}$ implies $\mathbf{d}(sk,c) = \mathbf{d}_{imp}(sk,c)$.
- If $(\langle \mathfrak{d}, (sk,c)\rangle, b) \in \mathsf{Freq}$ for some bit b, then by the way in which Freq is sampled we have $(\langle \mathfrak{e}, (pk,b,*)\rangle, c) \in \mathsf{Freq}$. Now since $\mathsf{Q_c}$ should agree with Freq it holds either that (a) $(\langle \mathfrak{e}, (pk,*,*)\rangle, c) \notin \mathsf{Q_c}$ or (b) if $(\langle \mathfrak{e}, (pk,b',*)\rangle, c) \in \mathsf{Q_c}$ then $b' = b$. Both (a) and (b) imply $\mathbf{d}_{imp}(sk,c) = b$, and thus $\mathbf{d}(sk,c) = \mathbf{d}_{imp}(sk,c)$.

The proof of Lemma 11 is now complete. □

Lemma 5, Part (B) (Restated). *Fix $(\mathbf{O}, \mathbf{u}, \mathbf{w}, PK) \in Env(n)$ and let $M = 0^n 1^n$. 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 $1 \leq i \leq 2n^{\vartheta+1}$*

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

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

Proof. The proof of this part of the lemma follows very similarly to that of Part (A), so we will omit details whenever appropriate.

Fix $i \in \{1, \ldots, 2n^{\vartheta+1}\}$ as in the lemma. All probabilities below are taken over the random coins described in the lemma, unless otherwise stated. Define the random variable $(s\mathcal{k}, c)$ as follows: $(s\mathcal{k}, c) = (sk_i, c_i)$ if $qu_i = \langle \mathbf{d}, (sk_i, c_i)\rangle$, and $(s\mathcal{k}, c) = (\bot, \bot)$, otherwise. We re-write the probability of the theorem as

$$\beta(n) = \Pr[(\underbrace{(s\mathcal{k}, c) \neq (\bot, \bot)}_{Evnt1}) \wedge (\underbrace{\mathbf{d}(s\mathcal{k}, c) \neq \widetilde{\mathbf{d}}(s\mathcal{k}, c)}_{Evnt2}) \wedge (\underbrace{\forall j < i, \mathbf{O}(qu_j) = \widetilde{\mathbf{O}}(qu_j)}_{Evnt3})].$$

We will show a stronger statement, showing

$$\beta(n) \leq \frac{1}{n^{10\vartheta}}.$$

Define $p\mathcal{K}$ to be $\mathbf{g}(s\mathcal{K})$ if $s\mathcal{K} \neq \perp$, and $p\mathcal{K} = \perp$, otherwise.

First note that if $Evnt1 \wedge Evnt2 \wedge Evnt3$ holds, since qu_i is the first query where $\mathbf{O}(qu_i) \neq \widetilde{\mathbf{O}}(qu_i)$, by Assumption 3 we have $\widetilde{\mathbf{g}}(s\mathcal{K}) = p\mathcal{K} \neq \perp$. Thus, by Lemma 11 we will have $(\langle \mathfrak{e}, (p\mathcal{K}, *, *)\rangle, c) \in \mathsf{Q_c}$, $(\langle \mathfrak{e}, (p\mathcal{K}, *, *)\rangle, c) \notin \mathsf{Freq}$ and $(\langle \mathfrak{d}, (s\mathcal{K}, c)\rangle, *) \notin \mathsf{Freq}$.

Fix an arbitrary ordering

$$((\langle \mathfrak{e}, (pk_1, b_1, r_1)\rangle, c_1), \ldots, (\langle \mathfrak{e}, (pk_m, b_m, r_m)\rangle, c_m))$$

on the elements of $\mathsf{Q_c}$. Note that $m \leq n^{\vartheta}$. Now by the discussion above if $Evnt1 \wedge Evnt2 \wedge Evnt3$ holds, then the following must hold:

– for some $1 \leq h \leq m$, $(p\mathcal{K}, c) = (pk_h, c_h)$, $(\langle \mathfrak{e}, (p\mathcal{K}, *, *)\rangle, c) \notin \mathsf{Freq}$ and $(\langle \mathfrak{d}, (s\mathcal{K}, c)\rangle, *) \notin \mathsf{Freq}$.

Thus, we have $\beta(n) \leq \sum_{h=1}^{m} \beta_h(n)$, where

$$\beta_h(n) = \Pr\left[\left((p\mathcal{K}, c) = (pk_h, c_h)\right) \wedge \left(\langle \mathfrak{e}, (p\mathcal{K}, *, *)\rangle, c) \notin \mathsf{Freq}\right) \wedge (\langle \mathfrak{d}, (s\mathcal{K}, c)\rangle, *) \notin \mathsf{Freq})\right].$$

We now claim

Claim I. For all $1 \leq h \leq m$

$$\beta_h(n) \leq \frac{1}{n^{10\vartheta}}.$$

Claim I implies the desired bound:

$$\beta(n) \leq m \times \frac{1}{n^{10\vartheta}} \leq \frac{1}{n^{8\vartheta}}.$$

Thus, we focus on proving Claim I. Fix $h \in \{1, \ldots, m\}$ in the sequel.

Call a pair (pk, c), with $sk = \mathbf{g}^{-1}(pk)$, *heavy* if for some $b \in \{0, 1\}$

$$\Pr_{R}\left[(\langle \mathbf{e}, (pk, *, *)\rangle, c) \text{ or } (\langle \mathbf{d}, (sk, c)\rangle, *) \text{ is made during } E^{\mathbf{O}}(PK, b; R)\right] \geq \frac{1}{n^{11\vartheta}}. \tag{41}$$

Note that in the equation above it may be that $sk = \perp$, in which case the query $\langle \mathbf{d}, (sk, c)\rangle$ is invalid, and so it is never asked, and thus the above probability amounts to the probability of the former condition alone. We call (pk, c) *unheavy* if the above probability is strictly less than $\frac{1}{n^{11\vartheta}}$ for both $b = 0$ and $b = 1$.

Recalling h that was fixed above, letting $sk_h = \mathbf{g}^{-1}(pk_h)$, to bound $\beta_h(n)$ we define the events Ev_1 and Ev_2:

1. Ev_1: there exists (pk, c) such that (pk, c) is heavy and $(\langle \mathfrak{e}, (pk, *, *)\rangle, c) \notin \mathsf{Freq}$ and $(\langle \mathfrak{d}, (\mathbf{g}^{-1}(pk), c)\rangle, *) \notin \mathsf{Freq}$
2. Ev_2: $(p\mathcal{K}, c) = (pk_h, c_h)$, $(\langle \mathbf{e}, (p\mathcal{K}, *, *)\rangle, c) \notin \mathsf{Freq}$, $(\langle \mathbf{d}, (s\mathcal{K}, c)\rangle, *) \notin \mathsf{Freq}$ and (pk_h, c_h) is unheavy.

We have

$$\beta_h(n) \leq \Pr[Ev_1] + \Pr[Ev_2].$$

We will show

$$\epsilon(n) \stackrel{\text{def}}{=} \Pr[Ev_1] \leq \frac{1}{n^{11\vartheta}};$$

and

$$\beta_{h,1}(n) \stackrel{\text{def}}{=} \Pr[Ev_2] \leq \frac{1}{n^{11\vartheta}}.$$

The above bounds for $\epsilon(n)$ and $\beta_{h,1}(n)$ imply the bound for $\beta_h(n)$, as desired.

Bounding $\epsilon(n)$. The proof consists of two parts. First, we show there are at most a polynomial number of heavy pairs (pk, c) (**Claim A**). Next, we show for a specific heavy pair (pk, c) the probability that

$$(\langle \mathfrak{e}, (pk, *, *)\rangle, c) \notin \mathsf{Freq} \wedge (\langle \mathfrak{d}, (\mathbf{g}^{-1}(pk), c)\rangle, *) \notin \mathsf{Freq}$$

is negligible (**Claim B**).

Proof of Claim A. Fix $b \in \{0,1\}$. We show the number of pairs (pk, c) for which Equation 41 holds for the fixed b is at most polynomial. Call (pk, c), with $sk = \mathbf{g}^{-1}(pk)$, *i-heavy* if during a random execution of $E^{\mathbf{O}}(PK, b; R)$ with probability at least $\frac{1}{n^{12\vartheta}}$ (taken over R) the ith query/response pair is either $(\langle \mathbf{e}, (pk, *, *)\rangle, c)$ or $(\langle \mathbf{d}, (sk, c)\rangle, *)$. If at least one of $(\langle \mathbf{e}, (pk, *, *)\rangle, c)$ or $(\langle \mathbf{d}, (sk, c)\rangle, *)$ is asked with probability at least $\frac{1}{n^{11\vartheta}}$ during $E^{\mathbf{O}}(PK, b; R)$ then for some $1 \le i \le n^{\vartheta}$ (pk, c) is i-heavy. For every i we have at most $n^{12\vartheta}$ i-heavy pairs. Thus, for a fixed b we have at most $n^{12\vartheta} \times n^{\vartheta}$ pairs for which Equation 41 holds. Thus, we have at most $2n^{13\vartheta}$ heavy pairs (pk, c), and the proof of Claim A is complete. □

Proof of Claim B. Recall that Freq is formed by collecting all query/response pairs made during $n^{23\vartheta}$ random executions of $E^{\mathbf{O}}(PK, 01)$. For $1 \le k \le n^{23\vartheta}$ let $x_k = 1$ if at least one of $(\langle \mathbf{e}, (pk, *, *)\rangle, c)$ and $(\langle \mathbf{d}, (sk, c)\rangle, *)$ is asked during the kth execution, and $x_k = 0$, otherwise. We know that $x_1, \ldots, x_{n^{23\vartheta}}$ are all identically distributed and independent, and that $p \stackrel{\text{def}}{=} \Pr[x_1 = 1] \ge \frac{1}{n^{11\vartheta}}$. Let $x_{av} = \frac{x_1 + \cdots + x_{n^{23\vartheta}}}{n^{23\vartheta}}$. We have

$$\begin{aligned}
&\Pr[(\langle \mathfrak{e}, (pk, *, *)\rangle, c) \notin \mathsf{Freq}) \wedge (\langle \mathfrak{d}, (sk, c)\rangle, *) \notin \mathsf{Freq}) \mid (pk, c) \text{ is heavy}] \\
&\le \Pr[|x_{av} - p| \ge p] \le \Pr[|x_{av} - p| \ge \frac{1}{n^{11\vartheta}}] \le \frac{1}{2^{2n^{23\vartheta - 22\vartheta}}} \le \frac{1}{2^{2n}},
\end{aligned}$$

where the third inequality above follows by Theorem 8. The proof of Claim B is complete. □

Bounding $\beta_{h,1}(n)$.

$$\begin{aligned}
\beta_{h,1}(n) &\le \Pr\left[\left((pk, c) = (pk_h, c_h)\right) \wedge ((pk_h, c_h) \text{ is unheavy})\right] \\
&\le \Pr\left[(pk, c) = (pk_h, c_h) \mid (pk_h, c_h) \text{ is unheavy}\right] \le \frac{1}{n^{11\vartheta}}.
\end{aligned}$$

The reason for the last inequality above is that since the index i, to which (pk, c) corresponds, is fixed, the probability that $(pk, c) = (pk_h, c_h)$ is at most the probability that at least one of $(\langle \mathbf{e}, (pk_h, *, *)\rangle, c_h)$ or $(\langle \mathbf{d}, (sk_h, c_h)\rangle, *)$ is made during a random encryption $E^{\mathbf{O}}(PK, b)$: Here b is the bit of $0^n 1^n$ to which the index i corresponds, and b is uniquely determined by Assumption 1.

The proof of Part (B) of Lemma 5 is now complete. □

C Proof of Lemma 7

Lemma 7, (Restated). Fix

$$(\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{u}, \mathbf{w}, PK) \in Env(n) \text{ and } (\mathsf{Freq}, SK', \mathsf{Q_s}, \mathsf{Q_c}, \widetilde{\mathbf{g}}, \widetilde{\mathbf{e}}, \widetilde{\mathbf{d}}) \in Tvars^{\mathcal{O}}(PK).$$

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

$$\mathsf{W}' = \{(pk_1, \mathbf{e}(pk_1, b_1, r_1)), \ldots, (pk_p, \mathbf{e}(pk_p, b_p, r_p))\}$$

and

$$\mathsf{W} = \{(pk_1, c_1), \ldots, (pk_p, c_p)\}.$$

Note that in the definition of W' by $\mathbf{e}(pk_i, b_i, r_i)$ we mean the actual value $\mathbf{e}(pk_i, b_i, r_i)$, not the notation itself.

All the following conditions hold.

(a) Both $\widetilde{\mathbf{g}}$ and $\widetilde{\mathbf{e}}$ can be computed efficiently (on all points) given access to the 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 the 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. First of all, recall that

$$(\mathbf{e}_{imp}, \mathbf{d}_{imp}) = EncImpose(\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathsf{Q_c})$$
$$(\widetilde{\mathbf{g}}, \widetilde{\mathbf{d}}) = KeyImpose(\mathbf{g}, \mathbf{e}_{imp}, \mathbf{d}_{imp}, \mathsf{Q_s})$$
$$\widetilde{\mathbf{e}} = \mathbf{e}_{imp}.$$

Part (a). The proof of this case follows easily by inspection. We give the proof for $\widetilde{\mathbf{e}}$, since this is the more difficult case. To compute $\widetilde{\mathbf{e}}(pk, b, r)$: if $(\langle \mathfrak{e}, (pk, b, r)\rangle, c) \in \mathsf{Q_c}$ for some c, then $\widetilde{\mathbf{e}}(pk, b, r) = c$; else if $(pk, \mathbf{e}(pk, b, r)) \notin \mathsf{W}$ then $\widetilde{\mathbf{e}}(pk, b, r) = \mathbf{e}(pk, b, r)$; else $\widetilde{\mathbf{e}}(pk, b, r) = \mathbf{e}(pk, b, r + x)$ where x is the smallest integer such that $(pk, \mathbf{e}(pk, b, r + x)) \notin \mathsf{W} \cup \mathsf{W}'$.

Part (b). If $(\langle \mathfrak{g}, sk\rangle, *) \notin \mathsf{Q_s}$ then by Claim 14 we have $\widetilde{\mathbf{d}}(sk, c) = \mathbf{d}_{imp}(sk, c)$. To compute $\mathbf{d}_{imp}(sk, c)$, first compute $pk = \mathbf{g}(sk)$; if $(\langle \mathfrak{e}, (pk, b, *)\rangle, c) \in \mathsf{Q_c}$ for some bit b, then $\mathbf{d}_{imp}(sk, c) = b$; otherwise, by definition $\mathbf{d}_{imp}(sk, c) = \mathbf{d}(sk, c)$.

Part (c). If for some pk, $(\langle \mathfrak{g}, sk\rangle, pk) \in \mathsf{Q_s}$, then by definition $\widetilde{\mathbf{d}}(sk, c)$ is computed as follows: if for some b and r it holds that $c = \mathbf{e}_{imp}(pk, b, r)$ then $\widetilde{\mathbf{d}}(sk, c) = b$; otherwise, $\widetilde{\mathbf{d}}(sk, c) = \perp$. By the assumption of this part of the lemma we have $(pk, c) \notin \mathsf{W} \cup \mathsf{W}'$. If $\mathbf{u}(pk, c) = \perp$ we claim that for no b and r does it hold that $c = \mathbf{e}_{imp}(pk, b, r)$; the reason is that since $(pk, c) \notin \mathsf{W}$ if $c = \mathbf{e}_{imp}(pk, b, r)$ then for some r' it holds $\mathbf{e}(pk, b, r') = c$ (this can be easily verified from the definition of $\mathbf{e}_{imp}$), which contradicts $\mathbf{u}(pk, c) = \perp$. Thus, $\widetilde{\mathbf{d}}(sk, c) = \perp$. On the other hand, if $\mathbf{u}(pk, c) = (b, *) \neq \perp$, then we know for some r we have $\mathbf{e}(pk, b, r) = c$. We claim $\mathbf{e}_{imp}(pk, b, r) = c$. Assume not: then by definition for some $i \leq p$ it holds that either $(pk, b, r) = (pk_i, b_i, r_i)$ or $(pk, \mathbf{e}(pk, b, r)) = (pk_i, c_i)$. The former implies $(pk, c) \in \mathsf{W}'$ and the latter implies $(pk, c) \in \mathsf{W}$, both contradicting the assumption $(pk, c) \notin \mathsf{W} \cup \mathsf{W}'$.

Part (d). If for some pk, $(\langle \mathfrak{g}, sk\rangle, pk) \in \mathsf{Q_s}$, then, again by definition, $\widetilde{\mathbf{d}}(sk, c) = b$ iff there exists b and r such that $c = \mathbf{e}_{imp}(pk, b, r)$. Since $(pk, c) \in \mathsf{W}'$ we know $pk = pk_i$ and $c = \mathbf{e}(pk_i, b_i, r_i)$ for some $i \leq p$. Assume toward a contradiction that for some b and r it holds that $c = \mathbf{e}_{imp}(pk, b, r)$. Thus, $\mathbf{e}_{imp}(pk, b, r) = c = \mathbf{e}(pk_i, b_i, r_i)$. We consider all possible cases, and prove each yields a contradiction:

- $(pk, b, r) = (pk_j, b_j, r_j)$ for some $j \leq p$: impossible since otherwise $\mathbf{e}_{imp}(pk, b, r) = c_j$ and so $(pk, c) = (pk_j, c_j) \in \mathsf{W}$.
- $(pk, \mathbf{e}(pk, b, r)) \notin \mathsf{W}'$: then since by Part (a) we know for no $j \leq p$ $(pk, b, c) = (pk_j, b_j, c_j)$, by applying the definition of $\mathbf{e}_{imp}$ we obtain $\mathbf{e}_{imp}(pk, b, r) = \mathbf{e}(pk, b, r)$. Since from earlier we have $\mathbf{e}_{imp}(pk, b, r) = \mathbf{e}(pk_i, b_i, r_i)$ and $pk = pk_i$, we obtain $(pk, \mathbf{e}(pk, b, r)) = (pk_i, \mathbf{e}(pk_i, b_i, r_i)) \in \mathsf{W}'$, which is a contradiction to the assumption that $(pk, \mathbf{e}(pk, b, r)) \notin \mathsf{W}'$.
- $(pk, \mathbf{e}(pk, b, r)) \in \mathsf{W}'$: Since by Part (a) we know for no $h \leq p$ it holds that $(pk, b, c) = (pk_h, b_h, c_h)$, by applying the definition of $\mathbf{e}_{imp}$ we obtain $\mathbf{e}_{imp}(pk, b, r) = \mathbf{e}(pk, b, r+x)$ for some x for which in particular $(pk, b, r + x) \neq (pk_i, b_i, r_i)$. Since from earlier we have $pk = pk_i$, we obtain $(b, r + x) \neq (b_i, r_i)$. Thus, we have $\mathbf{e}(pk, b, r + x) = c = \mathbf{e}(pk, b_i, r_i)$, which is a contradiction since $\mathbf{e}(pk, \cdot, \cdot)$ is one-to-one.

Part (e). If for some pk, $(\langle \mathfrak{g}, sk\rangle, pk) \in \mathsf{Q_s}$, then by definition $\widetilde{\mathbf{d}}(sk, c)$ is computed as follows: if there exists b and r such that $\mathbf{e}_{imp}(pk, b, r) = c$ then $\widetilde{\mathbf{d}}(sk, c) = b$; otherwise, $\widetilde{\mathbf{d}}(sk, c) = \perp$. Now since $(pk, c) = (pk_i, c_i)$, we have $\mathbf{e}_{imp}(pk, b_i, r_i) = c$ and so $\widetilde{\mathbf{d}}(sk, c) = b_i$.

The proof of Lemma 7 is now complete. □

D Proof of Lemma 9

We first start with the following notation.

Notation 1. $Textvars^{\mathcal{O}}(PK_1, \ldots, PK_{t+1})$ denotes the random variables

$$(\mathsf{Freq_i}, \mathsf{FreqPub_i}, \widetilde{SK}_i, \mathsf{Q^i_s}, \mathsf{Q^i_c}, \mathsf{QPub_i}, \widetilde{\mathbf{O}}_i)_{1 \le i \le t+1}$$

obtained in the execution of $\mathbf{T_{t+1}}(1^n, PK_1, \ldots, PK_{t+1}, \ldots)$ w.r.t. the oracle $\mathcal{O}$. Note that none of these random variables depend on $(C_1, \ldots, C_{t+1})$.

D.1 Proof of Lemma 9, Part 1

Proof. All probabilities that appear below are taken over the variables sampled as in the lemma. For any two fixed and distinct $\varsigma, \omega \in [t+1]$ we show

$$\alpha(n) \stackrel{\text{def}}{=} \Pr[(\mathsf{QPub}_\varsigma \cap \mathsf{QPub}_\omega) \setminus (\mathsf{FreqPub}_\varsigma \cup \mathsf{FreqPub}_\omega) \neq \emptyset] \le \frac{1}{n^{2t_0+9}}; \tag{42}$$

A union bound then yields the bound of the theorem.

Define

$$\mathsf{QPubReal}_\varsigma = \{pk \mid \text{ the query/response } (\langle \mathbf{g}, * \rangle, pk) \text{ appears during } G^{\mathbf{O}}(S_\varsigma)\}.$$

Define $\mathsf{QPubReal}_\omega$ similarly by replacing ς with ω in the equation above.

In what follows, we show that with all but negligible probability

$$\mathsf{QPub}_\varsigma \subseteq \mathsf{QPubReal}_\varsigma \cup \mathsf{FreqPub}_\varsigma \text{ and } \mathsf{QPub}_\omega \subseteq \mathsf{QPubReal}_\omega \cup \mathsf{FreqPub}_\omega.$$

This will allow us to focus our attention on bounding the probability of the event

$$(\mathsf{QPubReal}_\varsigma \cap \mathsf{QPubReal}_\omega) \setminus (\mathsf{FreqPub}_\varsigma \cup \mathsf{FreqPub}_\omega) \neq \emptyset.$$

In the following define the event `subset` as

$$\texttt{subset} \stackrel{\text{def}}{=} (\mathsf{QPub}_\varsigma \subseteq \mathsf{QPubReal}_\varsigma \cup \mathsf{FreqPub}_\varsigma) \wedge (\mathsf{QPub}_\omega \subseteq \mathsf{QPubReal}_\omega \cup \mathsf{FreqPub}_\omega).$$

Lemma 12.

$$\Pr[\texttt{subset}] \ge 1 - \frac{1}{2^n}.$$

Lemma 13.

$$\beta(n) \stackrel{def}{=} \Pr[(\mathsf{QPubReal}_\varsigma \cap \mathsf{QPubReal}_\omega) \setminus (\mathsf{FreqPub}_\varsigma \cup \mathsf{FreqPub}_\omega) \neq \emptyset] \le \frac{1}{2n^{2t_0+9}}. \tag{43}$$

In what follows we will first show that Lemma 9, Part 1 follows easily from Lemmas 12 and 13. Then, we will prove each of the lemmas.

Let $\alpha(n)$ be as above. We have

$$\begin{aligned}
\alpha(n) &\le \Pr[((\mathsf{QPub}_\varsigma \cap \mathsf{QPub}_\omega) \setminus (\mathsf{FreqPub}_\varsigma \cup \mathsf{FreqPub}_\omega) \neq \emptyset) \wedge \texttt{subset}] + \Pr[\overline{\texttt{subset}}] \\
&\le \Pr[(\mathsf{QPubReal}_\varsigma \cap \mathsf{QPubReal}_\omega) \setminus (\mathsf{FreqPub}_\varsigma \cup \mathsf{FreqPub}_\omega) \neq \emptyset] + \frac{1}{2^n} \\
&\le \frac{1}{2n^{2t_0+9}} + \frac{1}{2^{2n}} \le \frac{1}{n^{2t_0+9}}.
\end{aligned}$$

Proof of Lemma 12. Let $\mathsf{subset}_\varsigma$ be the event $(\mathsf{QPub}_\varsigma \subseteq \mathsf{QPubReal}_\varsigma \cup \mathsf{FreqPub}_\varsigma)$ and subset_ω be the event $(\mathsf{QPub}_\omega \subseteq \mathsf{QPubReal}_\omega \cup \mathsf{FreqPub}_\omega)$. We show $\Pr[\overline{\mathsf{subset}_\varsigma}] \leq \frac{1}{2^{2n}}$. Using the same argument we can show $\Pr[\overline{\mathsf{subset}_\omega}] \leq \frac{1}{2^{2n}}$, establishing the desired bound.

The idea of the proof is as follows: We show how to successfully forge a public key $pk \in \{0,1\}^{5n}$ by making a polynomial number of queries whenever $\overline{\mathtt{subset}_\varsigma}$ holds. Then by relying on Lemma 2 we obtain $\Pr[\overline{\mathsf{subset}_\varsigma}] \leq \frac{1}{2^{2n}}$. The forgery attack is enabled by the fact that during the formations of QPub_ς from $(PK_\varsigma, \mathsf{FreqPub}_\varsigma)$ no real queries except except $\mathbf{w}$ queries are made. (See Step 2 of $\mathbf{T_{t+1}}$'s computation.) Details follow.

The forgery attack is done as follows: Sample $S_\varsigma \leftarrow \{0,1\}^n$, form

$$(*, PK_\varsigma) = G^{\mathbf{O}}(S_\varsigma)$$

and form Freq_ς and $\mathsf{FreqPub}_\varsigma$ and QPub_ς as in $\mathbf{T_{t+1}}$'s computation. That is,

$$\begin{aligned}
\mathsf{Freq}_\varsigma &\leftarrow ExtFreqQue^{\mathbf{O},\mathbf{u}}(PK_\varsigma, n^{23\vartheta+4t_0}) \\
\mathsf{FreqPub}_\varsigma &= \{pk \mid (\langle \mathfrak{g}, * \rangle, pk) \in \mathsf{Freq}_\varsigma\} \\
(\widetilde{SK}_\varsigma, \mathsf{Q}_\mathsf{s}^\varsigma, \mathsf{Q}_\mathsf{c}^\varsigma) &\leftarrow ConsOrc(PK_\varsigma, \mathsf{Freq}_\varsigma) \\
\mathsf{QPub}_\varsigma &= \{pk \mid \mathbf{w}(pk) = \top \text{ and } (\langle \mathfrak{g}, * \rangle, pk) \in \mathsf{Q}_\mathsf{s}^\varsigma\}.
\end{aligned}$$

Also, let

$$\mathsf{QPubReal}_\varsigma = \{pk \mid \text{ the query/response } (\langle \mathbf{g}, * \rangle, pk) \text{ appears during } G^{\mathbf{O}}(S_\varsigma)\}.$$

If there exists pk such that

$$pk \in \mathsf{QPub}_\varsigma \setminus (\mathsf{QPubReal}_\varsigma \cup \mathsf{FreqPub}_\varsigma)$$

then halt and return pk. Otherwise, return $\perp$.

Letting Que be the set of all query/response pairs made during this attack we show whenever the attacker returns $pk \neq \perp$ it holds that (I) $\mathbf{w}(pk) = \top$ and (II) $(\langle \mathbf{g}, * \rangle, pk) \notin \mathsf{Que}$ (which shows in these cases the attack is successful). Condition (I) obviously holds. To prove Condition (II) note that all $\mathbf{g}$-type query/response pairs in Que are made either during the execution of $G^{\mathbf{O}}(S_\varsigma)$ or during the formation of Freq_ς. By definition, if a query/response pair $(\langle \mathbf{g}, * \rangle, pk)$ is made during the execution $G^{\mathbf{O}}(S_\varsigma)$ then $pk \in \mathsf{QPubReal}_\varsigma$. Also, if a query/response pair $(\langle \mathbf{g}, * \rangle, pk)$ is made during the formation of Freq_ς then $pk \in \mathsf{FreqPub}_\varsigma$. Thus, if $pk \notin \mathsf{QPubReal}_\varsigma \cup \mathsf{FreqPub}_\varsigma$ then $(\langle \mathbf{g}, * \rangle, pk) \notin \mathsf{Que}$, as desired. The proof is complete. □

Proof of Lemma 13. Let $\widehat{pk_1}, \ldots, \widehat{pk_z}$, for some $z \leq n^\vartheta$ be the elements of $\mathsf{QPubReal}_\varsigma$. Note that $\widehat{pk_1}, \ldots, \widehat{pk_z}$ are all random variables. Toward bounding $\beta(n)$ (Equation 43) we define $\beta_i(n)$ for $1 \leq i \leq z$ as follows:

$$\beta_i(n) = \Pr[(\widehat{pk_i} \in \mathsf{QPubReal}_\omega) \wedge (\widehat{pk_i} \notin \mathsf{FreqPub}_\varsigma \cup \mathsf{FreqPub}_\omega)]. \tag{44}$$

We have $\beta(n) = \sum_{i \leq z} \beta_i(n)$. In what follows for an arbitrary fixed i we show

$$\beta_i(n) \leq \frac{1}{2n^{\vartheta+2t_0+9}}, \tag{45}$$

and then the claimed bound for $\beta(n)$ follows by the union bound.

Call pk heavy if

$$\Pr[\text{the query/response } (\langle \mathbf{g}, * \rangle, pk) \text{ appears during } G^{\mathbf{O}}(1^n)] \geq \frac{1}{n^{\vartheta+2t_0+10}}. \tag{46}$$

We call pk unheavy if pk is not heavy.

Using the notion of heaviness, we further break down $\beta_i(n)$ as follows.

$$\beta_{1,i}(n) = \Pr[(\widehat{pk_i} \in \mathsf{QPubReal}_\omega) \wedge (\widehat{pk_i} \notin \mathsf{FreqPub}_\varsigma \cup \mathsf{FreqPub}_\omega) \wedge (\widehat{pk_i} \text{ is unheavy })]$$
$$\beta_{2,i}(n) = \Pr[(\widehat{pk_i} \in \mathsf{QPubReal}_\omega) \wedge (\widehat{pk_i} \notin \mathsf{FreqPub}_\varsigma \cup \mathsf{FreqPub}_\omega) \wedge (\widehat{pk_i} \text{ is heavy })].$$

In the rest of the proof we show $\beta_{1,i}(n) \leq \frac{1}{4n^{\vartheta+2t_0+9}}$ and $\beta_{2,i}(n) \leq \frac{1}{4n^{\vartheta+2t_0+9}}$, which imply $\beta_i(n) \leq \frac{1}{2n^{\vartheta+2t_0+9}}$, as desired.

UpperBounding $\beta_{1,i}(n)$. We have

$$\begin{aligned}\beta_{1,i}(n) &\leq \Pr[(\widehat{pk_i} \in \mathsf{QPubReal}_\omega) \wedge (\widehat{pk_i} \text{ is unheavy })] \\ &\leq \Pr[(\widehat{pk_i} \in \mathsf{QPubReal}_\omega) \mid (\widehat{pk_i} \text{ is unheavy })] \leq^* \frac{1}{n^{\vartheta+2t_0+10}} \leq \frac{1}{4n^{\vartheta+2t_0+9}},\end{aligned}$$

as claimed. The inequality marked with $*$ follows from the definition of unheaviness and the way $\mathsf{QPubReal}_\omega$ is defined: the set of all pk where $(\langle \mathbf{g}, * \rangle, pk)$ appears during the execution of $G^{\mathbf{O}}(S_\omega)$, for $S_\omega \leftarrow \{0,1\}^n$.

UpperBounding $\beta_{2,i}(n)$. Just as above we have

$$\beta_{2,i}(n) \leq \epsilon(n) \stackrel{\text{def}}{=} \Pr[\widehat{pk_i} \notin \mathsf{FreqPub}_\varsigma \cup \mathsf{FreqPub}_\omega \mid \widehat{pk_i} \text{ is heavy }]. \tag{47}$$

Recall that during the formations of each of $\mathsf{FreqPub}_\varsigma$ and $\mathsf{FreqPub}_\omega$, among other executions, we execute $G^{\mathbf{O}}(1^n)$ independently $n^{23\vartheta+4t_0}$ times and record all pk for which we have a query/response $(\langle \mathbf{g}, * \rangle, pk)$ in the underlying set.

Thus, to bound $\epsilon(n)$ it suffices to bound the probability that, conditioned on $\widehat{pk_i}$ being heavy, that $(\langle \mathbf{g}, * \rangle, \widehat{pk_i}) \notin \mathsf{Q}$, where Q is formed by collecting all query/response pairs made during $n^{23\vartheta+4t_0}$ random executions of $G^{\mathbf{O}}(1^n)$. To this end, for $k \leq n^{23\vartheta+4t_0}$ define $x_k = 1$ if $(\langle \mathbf{g}, * \rangle, \widehat{pk_i})$ is made during the kth execution, and $x_k = 0$, otherwise. We know that $x_1, \ldots, x_{n^{23\vartheta+4t_0}}$ are all identically distributed and independent, and that $p \stackrel{\text{def}}{=} \Pr[x_1 = 1] \geq \frac{1}{n^{\vartheta+2t_0+10}}$. Let $x_{av} = (x_1 + \cdots + x_{n^{23\vartheta+4t_0}})/n^{23\vartheta+4t_0}$. We have

$$\begin{aligned}\epsilon(n) &\leq \Pr[(\langle \mathbf{g}, * \rangle, \widehat{pk_i}) \notin \mathsf{Q} \mid \widehat{pk_i} \text{ is heavy}] \leq \Pr[|x_{av} - p| \geq p] \leq \Pr[|x_{av} - p| \geq \frac{1}{n^{\vartheta+2t_0+10}}] \\ &\leq \frac{1}{2^{2n^{23\vartheta+4t_0-2\vartheta-4t_0-20}}} \leq \frac{1}{2^{2n}} \leq \frac{1}{4n^{\vartheta+2t_0+9}}.\end{aligned}$$

□

D.2 Proof of Lemma 9, Part 2

We will prove that

$$\Pr[S_1 \neq \widetilde{D^{\mathbf{O}_{t+1}}}(SK'_{t+1}, C_{t+1})] \leq \frac{1}{n^{6+t_0}}. \tag{48}$$

With exactly the same argument we can show that for any $2 \leq i \leq t+1$

$$\Pr[S_i \neq \widetilde{D^{\mathbf{O}_{i-1}}}(SK'_{i-1}, C_{i-1})] \leq \frac{1}{n^{6+t_0}}.$$

The claimed result then follows using a union bound.

Toward proving Equation 48 we need the following lemma.

Lemma 14. *Fix $(\mathbf{O}, \mathbf{u}, \mathbf{w}, PK_1, \ldots, PK_{t+1}) \in Env_{t+1}(n)$. Let $(qu_1, \ldots, qu_{n^\vartheta})$ denote the oracle queries asked during the execution of $E^{\mathbf{O}}(PK_{t+1}, S; R)$, for a random $S \leftarrow \{0,1\}^n$ and random R. Then, for any query index $1 \leq i \leq n^\vartheta$*

(A) $\Pr\left[(qu_i \textit{ is } \mathbf{g}\textit{- or } \mathbf{e}\textit{-type}) \wedge \left(\forall j < i, \mathbf{O}(qu_j) = \widetilde{\mathbf{O_{t+1}}}(qu_j)\right) \wedge \left(\mathbf{O}(qu_i) \neq \widetilde{\mathbf{O_{t+1}}}(qu_i)\right)\right] \leq \frac{1}{n^{8\vartheta+t_0}}$,

(B) $\Pr\left[(qu_i \textit{ is } \mathbf{d}\textit{-type}) \wedge \left(\forall j < i, \mathbf{O}(qu_j) = \widetilde{\mathbf{O_{t+1}}}(qu_j)\right) \wedge \left(\mathbf{O}(qu_i) \neq \widetilde{\mathbf{O}_{t+1}}(qu_i)\right)\right] \leq \frac{1}{n^{8\vartheta+t_0}}$,

where the probabilities are taken over $S \leftarrow \{0,1\}^n$*, random choice of* R *and*

$$(\mathsf{Freq_i}, \mathsf{FreqPub_i}, \widetilde{SK}_i, \mathsf{Q_s^i}, \mathsf{Q_c^i}, \mathsf{QPub_i}, \widetilde{\mathbf{O}}_i)_{1 \leq i \leq t+1} \leftarrow Textvars^{\mathcal{O}}(PK_1, \ldots, PK_{t+1}).$$

We first show how to use Lemma 14 to prove Equation 48. The proof is very similar to the the proof of Lemma 3 and so we omit most of the details.

Proof of Lemma 9, Part 2 (Equation 48). All the following probabilities are taken over the variables sampled in the lemma.

Let QS be the set of all queries asked during the execution under which $C_{t+1} \leftarrow E^{\mathbf{O}}(PK_{t+1}, S_1)$ was produced. As in the proof of Lemma 3 we have

$$\Pr[S_1 \neq D^{\widetilde{\mathbf{O}_{t+1}}}(SK'_{t+1}, C_{t+1})] \leq \beta(n) \stackrel{\text{def}}{=} \Pr\left[\exists qu \in \mathsf{QS}\colon \mathbf{O}(qu) \neq \widetilde{\mathbf{O_{t+1}}}(qu)\right].$$

To bound $\beta(n)$ we have

$$\beta(n) = \Pr\left[\exists qu \in \mathsf{QS}\colon \mathbf{O}(qu) \neq \widetilde{\mathbf{O_{t+1}}}(qu)\right] \leq n^{\vartheta} \times \frac{1}{n^{8\vartheta+t_0}} \leq \frac{1}{n^{6+t_0}}.$$

The first inequality above is obtained by applying Lemma 14 and a union bound. (Note that $|\mathsf{QS}| = n^{\vartheta}$.) The proof is now complete. □

The proof of Lemma 14 follows very similarly to that of Lemma 5, except for a few simple changes. In what follows, we show the proof of Part (A) of Lemma 14 and omit the proof of Part (B), as the latter follows similarly.

Proof of Lemma 14, Part (A). Fix $(\mathbf{O}, \mathbf{u}, \mathbf{w}, PK_1, \ldots, PK_{t+1})$ as in the lemma and let

$$\mathsf{QS} = \{qu_1, \ldots, qu_{n^{\vartheta}}\},$$

All probabilities, if not otherwise stated, are taken over the variables sampled in the lemma. We prove a stronger result, showing

$$\alpha(n) = \Pr\left[(\exists qu \in \mathsf{QS} \text{ such that } qu \text{ is } \mathbf{g}\text{- or } \mathbf{e}\text{-type}) \wedge \left(\mathbf{O}(qu) \neq \widetilde{\mathbf{O_{t+1}}}(qu)\right)\right] \leq \frac{1}{n^{8\vartheta+t_0}},$$

Let $\mathsf{DifQue_c}$ be formed as follows: for any $(\langle \mathfrak{e}, (pk, b, r)\rangle, c) \in \mathsf{Q_c^{t+1}}$:

(a) add $\langle \mathfrak{e}, (pk, b, r)\rangle$ to $\mathsf{DifQue_c}$, and
(b) if $\mathbf{u}(pk, c) = (b', r') \neq \perp$, also add $\langle \mathfrak{e}, (pk, b', r')\rangle$ to $\mathsf{DifQue_c}$.

By Lemma 6, $\mathsf{DifQue_c}$ is a superset of all $\mathfrak{e}$-type queries that receive different responses from $\mathbf{e}$ and $\widetilde{\mathbf{e}_{t+1}}$. Also, for any $(\langle \mathfrak{g}, sk\rangle, *) \in \mathsf{Q_s^{t+1}}$ add $\langle \mathfrak{g}, sk\rangle$ to the set $\mathsf{DifQue_s}$. Similarly, $\mathsf{DifQue_s}$ is a super-set of all $\mathfrak{g}$-type queries responded to differently under $\mathbf{g}$ and $\widetilde{\mathbf{g}_{t+1}}$. Fix an arbitrary ordering $(\langle \mathfrak{h}_1, q_1\rangle, \ldots, \langle \mathfrak{h}_m, q_m\rangle)$ on the elements of $\mathsf{DifQue_c} \cup \mathsf{DifQue_s}$, and note that $m \leq 2n^{\vartheta}$. As in the proof of Lemma 5, Part (A) we have

$$\alpha(n) \leq \beta(n) \stackrel{\text{def}}{=} \Pr\left[\text{for some } 1 \leq i \leq m\colon\ (\langle \mathbf{h}_i, q_i\rangle \in \mathsf{QS}) \wedge ((\langle \mathfrak{h}_i, q_i\rangle, *) \notin \mathsf{Freq_{t+1}})\right].$$

For $\mathbf{h} \in \{\mathbf{g}, \mathbf{e}\}$ call $\langle \mathbf{h}, q\rangle$ *heavy* if

$$\Pr_{S\leftarrow\{0,1\}^n,R}\left[\text{the query } \langle \mathbf{h}, q\rangle \text{ is asked during } E^{\mathbf{O}}(PK_{t+1}, S; R)\right] \geq \frac{1}{n^{11\vartheta+t_0}}. \tag{49}$$

We call $\langle \mathbf{h}, q\rangle$ *unheavy* if the above probability is strictly less than $\frac{1}{n^{11\vartheta+t_0}}$. Now defining

$$\epsilon(n) = \Pr\left[\exists \text{ query } \langle \mathbf{h}, q\rangle \text{ s.t. } (\langle \mathbf{h}, q\rangle \text{ is heavy }) \wedge (\langle \mathfrak{h}, q\rangle, *) \notin \mathsf{Freq_{t+1}})\right],$$
$$\beta_{1,i}(n) = \Pr\left[(\langle \mathbf{h}_i, q_i\rangle \in \mathsf{QS}) \wedge (\langle \mathfrak{h}_\mathfrak{i}, q_i\rangle \notin \mathsf{Freq_{t+1}}) \wedge (\langle \mathbf{h}_i, q_i\rangle \text{ is unheavy})\right],$$

we have $\beta(n) \leq \epsilon(n) + \sum_{1\leq i\leq m}(\beta_{1,i}(n))$. We claim

Claim I. $\beta_{1,i}(n) \leq \frac{1}{n^{11\vartheta+t_0}}$, for any fixed $1 \leq i \leq m$.

Claim II. $\epsilon(n)$ is negligible.

Claims I and II imply $\beta(n) \leq nel(n) + 2n^{\vartheta} \times \frac{1}{n^{11\vartheta+t_0}} \leq \frac{1}{n^{8\vartheta+t_0}}$, as desired. We now prove the two claims.

Proof of Claim I. Recall that QS contains queries made during $E(PK, S; R)$ for a random $S \leftarrow \{0,1\}^n$ and random R. We have

$$\beta_{1,i}(n) \leq \Pr\left[(\langle \mathbf{h}_i, q_i\rangle \in \mathsf{QS}) \wedge (\langle \mathbf{h}_i, q_i\rangle \text{ is unheavy})\right] \leq \Pr\left[\langle \mathbf{h}_i, q_i\rangle \in \mathsf{QS} \mid \langle \mathbf{h}_i, q_i\rangle \text{ is unheavy}\right]$$

and thus $\beta_{1,i}(n) \leq \frac{1}{n^{11\vartheta+t_0}}$, as claimed. □

Proof of Claim II. Exactly as in the proof of Lemma 5, Part (A) (Claim A, Page 23), we can show that there are at most a poly number of heavy queries. Thus, in what follows we show for a specific heavy query $\langle \mathbf{h}, q\rangle$ the probability that $(\langle \mathfrak{h}, q\rangle, *) \notin \mathsf{Freq_{t+1}}$ is negligible.

Recall that $\mathsf{Freq_{t+1}}$ is formed by collecting, among others, all query/response pairs made during $n^{23\vartheta+4t_0}$ random executions of $E^{\mathbf{O}}(PK_{t+1}, S)$, where in each execution S is chosen freshly at random from $\{0,1\}^n$. For $1 \leq k \leq n^{23\vartheta+4t_0}$ let $x_k = 1$ if $\langle \mathbf{h}, q\rangle$ is asked during the kth execution, and $x_k = 0$, otherwise. We know that $x_1, \ldots, x_{n^{23\vartheta+4t_0}}$ are all identically distributed and independent, and that $p \stackrel{\text{def}}{=} \Pr[x_1 = 1] \geq \frac{1}{n^{11\vartheta+t_0}}$. Let $x_{av} = (x_1 + \cdots + x_{n^{23\vartheta+4t_0}})/n^{23\vartheta+4t_0}$. We have

$$\Pr[(\langle \mathfrak{h}, q\rangle, *) \notin \mathsf{Freq} \mid \langle \mathbf{h}, q\rangle \text{ is heavy}] \leq \Pr[|x_{av} - p| \geq p] \leq \Pr[|x_{av} - p| \geq \frac{1}{n^{11\vartheta+t_0}}].$$

By Theorem 8 the last probability above is at most $\frac{1}{2^{2n^{23\vartheta+4t_0-22\vartheta-2t_0}}} \leq \frac{1}{2^{2n}}$. □

□

E Summary of variables inside $Tvars$

$\mathsf{Freq} \leftarrow FreqQue^{\mathbf{O},\mathbf{u}}(PK, n^{23\vartheta})$ is sampled as follows.

1. For $b = 0, 1$: run $E^{\mathbf{O}}(PK, b)$ $n^{23\vartheta}$ times and add the symbolic versions of all query/response pairs to Freq.
2. 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.

$(SK', \mathsf{Q_s}, \mathsf{Q_c}) \leftarrow ConsOrc(PK, \mathsf{Freq})$ is sampled as follows.

1. Sample $(\mathbf{g}', \mathbf{e}', \mathbf{d}', S')$ under the constraints that $\mathbf{O}' = (\mathbf{g}', \mathbf{e}', \mathbf{d}')$ is Ψ-valid and is consistent with Freq and that $G^{\mathbf{O}'}(S') = (*, PK)$.
2. Let SK' be the secret-key output of $G^{\mathbf{O}'}(S')$ and let $\mathsf{Q_s}$ and $\mathsf{Q_c}$ contain, respectively, the symbolic versions of all the query/response pairs made to $\mathbf{g}'$ and $\mathbf{e}'$.

$(\mathbf{e}_{imp}, \mathbf{d}_{imp}) = EncImpose(\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathsf{Q_c})$ is constructed as follows.

1. Assuming that $\mathsf{Q_c} = \{(\langle \mathfrak{e}, (pk_1, b_1, r_1)\rangle, c_1), \ldots, (\langle \mathfrak{e}, (pk_p, b_p, r_p)\rangle, c_p)\}$ let $\mathsf{W} = \{(pk_1, c_1), \ldots, (pk_p, c_p)\}$.
2. Let $\mathsf{W}' = \{(pk_1, \mathbf{e}(pk_1, b_1, r_1)), \ldots, (pk_p, \mathbf{e}(pk_p, b_p, r_p))\}$
3. $$\mathbf{e}_{imp}(pk, b, r) = \begin{cases} c_i & \text{if } (pk, b, r) = (pk_i, b_i, r_i), \text{ for some } 1 \leq i \leq p \\ \hat{c} & \text{if } (pk, \mathbf{e}(pk, b, r)) \in \mathsf{W} \\ \mathbf{e}(pk, b, r) & \text{otherwise} \end{cases} \quad (50)$$
   where $\hat{c}$ is defined as follows: Letting x be the smallest integer where $(pk, \mathbf{e}(pk, b, r + x)) \notin \mathsf{W} \cup \mathsf{W}'$, we define $\hat{c} = \mathbf{e}(pk, b, r + x)$.
4. $$\mathbf{d}_{imp}(sk, c) = \begin{cases} b_i & \text{if } \mathbf{g}(sk) = pk_i \text{ and } c = c_i \text{ for some } 1 \leq i \leq p \\ \mathbf{d}(sk, c) & \text{otherwise} \end{cases} \quad (51)$$

Let $\widetilde{\mathbf{e}} = \mathbf{e}_{imp}$.The oracles $(\widetilde{\mathbf{g}}, \widetilde{\mathbf{d}}) = KeyImpose(\mathbf{g}, \mathbf{e}_{imp}, \mathbf{d}_{imp}, \mathsf{Q_s})$ are built as follows.

1. $$\widetilde{\mathbf{g}}(sk) = \begin{cases} \mathbf{g}(sk) & \text{if } sk \notin \{sk_1, \ldots, sk_w\} \\ pk_i & \text{if } sk = sk_i \text{ for some } 1 \leq i \leq w \end{cases} \quad (52)$$
2. $\widetilde{\mathbf{d}}(sk, c)$ 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, $\widetilde{\mathbf{d}}(sk, c) = \perp$.

Table 1. Summary of $Tvars$ variables w.r.t. PK and $(\mathbf{g}, \mathbf{e}, \mathbf{d}, \mathbf{u}, \mathbf{w})$