Selective-Opening Security in the Presence of Randomness Failures

Viet Tung Hoang[1] Jonathan Katz[2] Adam O'Neill[3] Mohammad Zaheri[4]

September 6, 2016

Abstract

We initiate the study of public-key encryption (PKE) secure against selective-opening attacks (SOA) in the presence of *randomness failures*, i.e., when the sender may (inadvertently) use low-quality randomness. In the SOA setting, an adversary can adaptively corrupt senders; this notion is natural to consider in tandem with randomness failures since an adversary may target senders by multiple means.

Concretely, we first treat SOA security of *nonce-based PKE*. After formulating an appropriate definition of SOA-secure nonce-based PKE, we provide efficient constructions in the non-programmable random-oracle model, based on lossy trapdoor functions.

We then lift our notion of security to the setting of "hedged" PKE, which ensures security as long as the sender's seed, message, and nonce *jointly* have high entropy. This unifies the notions and strengthens the protection that nonce-based PKE provides against randomness failures even in the non-SOA setting. We lift our definitions and constructions of SOA-secure nonce-based PKE to the hedged setting as well.

[1] Dept. of Computer Science, Florida State University, Email: `tvhoang@cs.fsu.edu`
[2] Dept. of Computer Science, University of Maryland, Email: `jkatz@cs.umd.edu`
[3] Dept. of Computer Science, Georgetown University, Email: `adam@cs.georgetown.edu`
[4] Dept. of Computer Science, Georgetown University, Email: `mz394@georgetown.edu`

Contents

1 Introduction

Imagine that an adversary wants to gain access to encrypted communication that various senders are transmitting to a receiver. There are various ways to go about doing this. One is to try to subvert the random-number generator used by the senders. Another is to break-in to the senders' machines, possibly in an adaptive fashion. Encryption schemes resisting the first sort of attack have been studied in the context of *security under randomness failures* [20, 3, 25, 7, 11] while resistance to the second sort of attack corresponds to the notion of *security against selective-opening attacks* (SOA) [9, 5, 17, 16, 12, 18].[1] However, as far as we are aware, these notions have so far only been considered separately. We initiate the study of SOA-secure encryption in the presence of randomness failures, providing new definitions and constructions achieving these definitions in the public-key setting.

There are currently three main approaches in the literature to dealing with randomness failures for PKE: (1) *deterministic* PKE [2], which does not use randomness at all but guarantees security only if plaintexts have high entropy, (2) *hedged* PKE, which is randomized and guarantees security as long as plaintexts and the randomness *jointly* have high entropy, and (3) the recently introduced notion of *nonce-based* PKE by Bellare and Tackmann (BT) [11], where each sender uses a uniform seed[2] in addition to a nonce, and security is guaranteed if either the seed is secret and the nonces are unique, or the seed is revealed and the nonces have high entropy. Hedged PKE and nonce-based PKE are incomparable and are useful in different scenarios, and part of our contribution is to unify them into a single primitive. We start by adding consideration of SOA security to nonce-based PKE. We then lift the resulting notions to the setting of hedged PKE (which subsumes deterministic PKE) as well, thereby adding consideration of SOA to a unified primitive with the guarantees of both nonce-based and hedged PKE.

1.1 Our Results

Selective-opening security for nonce-based PKE. As explained above, the first notion we consider for protecting against randomness failures is *nonce-based* PKE, recently introduced by Bellare and Tackmann [11]. For consistency with the definitions of SOA security we introduce for later notions (where new technical challenges arise), we formulate an indistinguishability-based (rather than simulation-based) definition, which we call N-SO-CPA, along the lines of the indistinguishability-based definition of SOA security for standard PKE [9]. Under our definition, the adversary can (i) learn the seeds of some senders, (ii) choose the nonces for all the other senders, as long as nonces of each individual sender do not repeat. Then, *after* seeing the ciphertexts, the adversary can adaptively corrupt some senders to learn their messages *together with* seeds and nonces. The definition asks that the adversary cannot distinguish between the plaintexts of the uncorrupted senders and a *resampling* of these plaintexts conditioned on the revealed plaintexts.

The next question is whether N-SO-CPA security is achievable. Throughout our work, we focus on constructions in the so-called non-programmable random-oracle model (NPROM) [22]. Intuitively, this means that in a security proof, the constructed adversary must honestly answer (i.e., cannot program) the random oracle queries of the assumed adversary. The NPROM is arguably closer to the standard (random oracle devoid) model than the programmable random oracle model (PROM), since real-world hash functions are not programmable. In this model, we give an efficient construction of N-SO-CPA-secure[3] nonce-based PKE based on any lossy trapdoor function [23]. The idea is to modify the nonce-based PKE scheme of Bellare and Tackmann, which encrypts a message m using public-key pk, seed xk, and nonce N by encrypting m using any standard (randomized) PKE scheme with public key pk and "synthetic" coins derived from a hash of (xk, N, m). Here, we use a *specific* randomized encryption scheme based on any lossy trapdoor function. The security proof of the resulting scheme, which we call $\mathsf{NE1}$, relies on switching to the lossy key-generation algorithm and then using the random oracle to argue that the adversary's choice of which senders to corrupt must be independent of the plaintexts.

SOA+hedged security for nonce-based PKE. Unlike nonce-based PKE, *hedged* PKE [3] guarantees security as long as the message and randomness used by the sender *jointly* have high entropy. Indeed, viewing the sender's seed and nonce together as the sender's randomness, nonce-based PKE as defined in [11] *lacks* such a guarantee. To get the best of both worlds, we would like to add such a guarantee to nonce-based PKE. This strengthens the protection provided against randomness failures even in the absence of SOA; however, sticking with the main theme of this work, we aim to achieve it in the SOA setting as well. This leads to a definition that we call HN-SO-CPA, which incorporates both hedged and SOA security into the existing notion of nonce-based PKE.

[1]There are two forms of SOA security, called coin-revealing (corresponding to sender corruption) and key-revealing (corresponding to receiver corruption). This paper concerns the first one.

[2] The idea is that because a seed is chosen infrequently, it can be generated using high-quality randomness.

[3]In the main body of the paper we treat both CPA and CCA security. For simplicity, we do not discuss CCA here.

Modeling SOA in the hedged setting is technically challenging. Indeed, Bellare et al. [4] recently showed that a simulation-based notion of SOA security for deterministic PKE (which is a special case of hedged PKE) is impossible to achieve. They also noted that a natural indistinguishability-based definition is (for different reasons) trivially impossible to achieve, and left open the problem of defining a meaningful (yet achievable) definition. To that end, we introduce a novel "comparison-based" definition of SOA for nonce-based PKE, inspired by the comparison-based definition of SOA for deterministic PKE [2, 6] combined with the indistinguishability-based definition of SOA for standard PKE [9]. Roughly, the definition requires that the adversary cannot predict any function of all the plaintexts (i.e., including those of the uncorrupted senders) with much better probability than by computing the same function on a resampling of all the plaintexts conditioned on the revealed plaintexts. For technical reasons, HN-SO-CPA does not protect partial information about the messages depending on the public key, so we still require N-SO-CPA to hold in addition.

We provide two approaches for achieving HN-SO-CPA + N-SO-CPA-secure nonce-based PKE. The first is a generic transform inspired by the "randomized-then-deterministic" transform of [3] in the setting of hedged security. Namely, we propose a "Nonce-then-Deterministic" (NtD) transform in which one obtains a new nonce-based PKE scheme by composing an underlying nonce-based PKE scheme with a deterministic PKE scheme. We require that the underlying deterministic PKE scheme meet a corresponding special case of the HN-SO-CPA definition that we call D-SO-CPA, and achieve it via a scheme DE1 in the NPROM. Interestingly, the scheme DE1 is exactly the recent construction of Bellare and Hoang [7], except that they assume the hash function is UCE-secure [8] and achieve standard security (not SOA). Again, the analysis is quite involved and deals with subtleties neither present in SOA for randomized PKE nor in prior work on deterministic PKE. Alternatively, we show that the scheme NE1 directly achieves both HN-SO-CPA and N-SO-CPA in the NPROM.

Separation results. Finally, to justify our developing new schemes in the setting of selective-opening security in the presence of randomness failures rather than using existing ones, we show that the N-SO-CPA and D-SO-CPA are not implied by the standard notions (non-SOA) of nonce-based PKE [11] and D-PKE [2], respectively. Our counter-examples rely on the recent result of Hofheinz, Rao, and Wichs (HRW) [17] that separates IND-CCA security from SOA security for randomized PKE. We also show that N-SO-CPA does not imply HN-SO-CPA for nonce-based PKE, meaning the hedged security does strengthen the notion considered for nonce-based PKE in [11].

Open question. We leave obtaining standard-model (versus NPROM) schemes achieving our notions as an open question. Note that our NtD transform is in the standard model, so if we had standard-model instantiations of the underlying primitives we would get a standard-model HN-SO-CPA + N-SO-CPA-secure nonce-based PKE as well.

1.2 Organization

In contrast to the order in which we explained the results above, in the main body of the paper we first present our results on SOA security for deterministic PKE, then move to our results on SOA security for nonce-based PKE, and then finally present our results on hedged security for SOA-secure nonce-based PKE. This is because the results for deterministic PKE constitute the technical core of our work, and form a basis for the results that follow.

2 Preliminaries

Notation and conventions. An adversary is an algorithm or tuple of algorithms. All algorithms may be randomized and are required to be efficient unless otherwise indicated; we let PPT stand for "probabilistic, polynomial time." For an algorithm A we denote by $x \leftarrow\$ A(\cdots)$ the experiment that runs A on the elided inputs with uniformly random coins and assigns the output to x, and $x \leftarrow\$ A(\cdots;r)$ to denote the same experiment, but under the coins r instead of randomly chosen ones. If A is deterministic we denote this instead by $x \leftarrow A(\cdots)$. We let $[A(\cdots)]$ denote the set of all possible outputs of A when run on the elided arguments. If S is a finite set then $s \leftarrow\$ S$ denotes choosing a uniformly random element from S and assigning it to s. We denote by $\Pr[\,P(x) \;:\; \ldots\,]$ the probability that some predicate P is true of x after executing the elided experiment.

Let $\mathbb{N}$ denote the set of all non-negative integers. For any $n \in \mathbb{N}$ we denote by $[n]$ the set $\{1, \ldots, n\}$. For a vector $\mathbf{x}$, we denote by $|\mathbf{x}|$ its length (number of components) and by $\mathbf{x}[i]$ its i-th component. For a vector $\mathbf{x}$ of length n and any $I \subseteq [n]$, we denote by $\mathbf{x}[I]$ the vector of length $|I|$ such that $\mathbf{x}[I] = (\mathbf{x}[i])_{i \in I}$. For a string X, we let $|X|$ denote its length. For any integer $1 \le i \le j \le |X|$, we write $X[i]$ to denote the ith bit of X, and $X[i,j]$ the substring from the i-th to the j-th bit (inclusive) of X.

Public-key encryption. A *public-key encryption scheme* PKE with message space Msg is a tuple of algorithms $(\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$. The key-generation algorithm Kg on input 1^k outputs a public key pk and secret key sk. The encryption algorithm Enc on inputs a public key pk and message $m \in \mathsf{Msg}(k)$ outputs a ciphertext c. The deterministic decryption algorithm Dec on inputs a secret key sk and ciphertext c outputs a message m or $\perp$. We require that for all $(pk, sk) \in [\mathsf{Kg}(1^k)]$ and all $m \in \mathsf{Msg}(1^k)$, the probability that $\mathsf{Dec}(sk, (\mathsf{Enc}(pk, m)) = m$ is 1. We say PKE is *deterministic* if Enc is deterministic.

Lossy trapdoor function. A *lossy trapdoor function* [23] with domain LDom and range LRng is a tuple of algorithms $\mathsf{LT} = (\mathsf{LT.IKg}, \mathsf{LT.LKg}, \mathsf{LT.Eval}, \mathsf{LT.Inv})$ that work as follows. Algorithm $\mathsf{LT.IKg}$ on input a unary encoding of the security parameter 1^k outputs an "injective" evaluation key ek and matching trapdoor td. Algorithm $\mathsf{LT.LKg}$ on input 1^k outputs a "lossy" evaluation key lk. Algorithm $\mathsf{LT.Eval}$ on inputs an (either injective or lossy) evaluation key ek and $x \in \mathsf{LDom}(k)$ outputs $y \in \mathsf{LRng}(1^k)$. Algorithm $\mathsf{LT.Inv}$ on inputs a trapdoor td and a $y' \in \mathsf{LRng}(k)$ outputs $x' \in \mathsf{LDom}(k)$. We require the following properties.

Correctness: For all $k \in \mathbb{N}$ and any $(ek, td) \in [\mathsf{LT.IKg}(1^k)]$, it holds that $\mathsf{Inv}(td, \mathsf{LT.Eval}(ek, x)) = x$ for every $x \in \mathsf{LDom}(k)$.

Key indistinguishability: For every distinguisher D, the advantage

$$\mathbf{Adv}^{\mathrm{ltdf}}_{\mathsf{LT},D}(k) = \Pr\left[\, D(ek) \Rightarrow 1 \;:\; (ek, td) \leftarrow\!\!\$\; \mathsf{LT.IKg}(1^k) \,\right] - \Pr\left[\, D(lk) \Rightarrow 1 \;:\; lk \leftarrow\!\!\$\; \mathsf{LT.LKg}(1^k) \,\right]$$

is negligible.

Lossiness: The size of the co-domain of $\mathsf{LT.Eval}(lk, \cdot)$ is at most $|\mathsf{LRng}(k)|/2^{\tau(k)}$ for all $k \in \mathbb{N}$ and all $lk \in [\mathsf{LT.LKg}(1^k)]$. We call τ the *lossiness* of LT.

If the function $\mathsf{LT.Eval}(ek, \cdot)$ is a permutation for any $k \in \mathbb{N}$ and any $(ek, td) \in [\mathsf{LT.IKg}(1^k)]$ then we call LT a *lossy trapdoor permutation*. Both RSA and Rabin are lossy trapdoor permutations under appropriate assumptions [21, 24].

Message samplers. A *message sampler* $\mathcal{M}$ is a PPT algorithm that takes as input 1^k and a string $param \in \{0,1\}^*$, and outputs a vector $\mathbf{m}$ of messages and a vector $\mathbf{a}$ of the same length. Each $\mathbf{a}[i]$ is the auxiliary information that an adversary gains in addition to $\mathbf{m}[i]$, if it breaks into the machine of the sender of $\mathbf{m}[i]$. For example, if each $\mathbf{m}[i]$ is a signature of some string $\mathbf{x}[i]$, then the adversary may be able to obtain even $\mathbf{x}[i]$ in its break-in. We require that $\mathcal{M}$ be associated with functions $v(\cdot)$ and $n(\cdot)$ such that for any $param \in \{0,1\}^*$, for any $k \in \mathbb{N}$, and any $\mathbf{m} \in [\mathcal{M}(1^k, param)]$, we have $|\mathbf{m}| = v(k)$ and $|\mathbf{m}[i]| = n(k)$, for every $i \le |\mathbf{m}|$.

A message sampler $\mathcal{M}$ is (μ, d)-*entropic* if

- For any $k \in \mathbb{N}$, any $I \subseteq \{1, \dots, v(k)\}$ such that $|I| \le d$, any $param \in \{0,1\}^*$, and $(\mathbf{m}, \mathbf{a}) \leftarrow\!\!\$\; \mathcal{M}(1^k, param)$, conditioning on messages $\mathbf{m}[I]$ and their auxiliary information $\mathbf{a}[I]$ and $param$, each other message $\mathbf{m}[j]$ (with $j \in \{1, \dots, v(k)\} \backslash I$) must have conditional min-entropy at least μ. Note that here $(\mathbf{m}, \mathbf{a})$ is sampled independent of the set I.
- Messages $\mathbf{m}[1], \dots, \mathbf{m}[|\mathbf{m}|]$ must be distinct, for any $param \in \{0,1\}^*$ and any $\mathbf{m} \in [\mathcal{M}(1^k, param)]$.

In this definition d can be ∞, which corresponds to a message sampler in which the conditional distribution of each message, given $param$ and all other messages and their corresponding auxiliary information, has at least μ bits of min-entropy.

Resampling. Following [9], let $\mathsf{Coins}[k]$ be the set of coins for $\mathcal{M}(1^k, \cdot)$, and $\mathsf{Coins}[k, \mathbf{m}^*, \mathbf{a}^*, I, param] = \{\omega \in \mathsf{Coins}[k] \mid \mathbf{m}'[I] = \mathbf{m}^* \text{ and } \mathbf{a}'[I] = \mathbf{a}^*, \text{where } (\mathbf{m}', \mathbf{a}') \leftarrow \mathcal{M}(1^k, param; \omega)\}$. Let $\mathsf{Resamp}_{\mathcal{M}}(1^k, I, \mathbf{m}^*, \mathbf{a}^*, param)$ be the algorithm that first samples $r \leftarrow\!\!\$\; \mathsf{Coins}[k, \mathbf{m}^*, \mathbf{a}^*, I, param]$, then runs $(\mathbf{m}', \mathbf{a}') \leftarrow \mathcal{M}(1^k, param; r)$, and then returns $\mathbf{m}'$. (Note that $\mathsf{Resamp}_{\mathcal{M}}$ may run in exponential time.) A *resampling algorithm* of $\mathcal{M}$ is an algorithm Rsmp such that $\mathsf{Rsmp}(1^k, I, \mathbf{m}^*, \mathbf{a}^*, param)$ and $\mathsf{Resamp}_{\mathcal{M}}(1^k, I, \mathbf{m}^*, \mathbf{a}^*, param)$ are identically distributed.[4] A message sampler $\mathcal{M}$ is *fully resamplable* if it admits a PPT resampling algorithm.

Partial resampling. We also introduce a new notion of "partial resampling." Let δ be a function and let $\mathsf{Resamp}_{\mathcal{M},\delta}(1^k, I, \mathbf{m}^*, \mathbf{a}^*, param)$ be the algorithm that samples $r \leftarrow\!\!\$\; \mathsf{Coins}[k, \mathbf{m}^*, \mathbf{a}^*, I, param]$, runs $(\mathbf{m}', \mathbf{a}') \leftarrow$

[4] Here for simplicity, we only consider $\mathcal{M}$ and Rsmp such that the distributions of $\mathsf{Rsmp}(1^k, I, \mathbf{m}^*, \mathbf{a}^*, param)$ and $\mathsf{Resamp}_{\mathcal{M}}(1^k, I, \mathbf{m}^*, \mathbf{a}^*, param)$ are identical. Following [9], one might also consider $\mathcal{M}$ and Rsmp such that the two distributions above are statistically close.

$\mathcal{M}(1^k, param; r)$, and then returns $\delta(\mathbf{m}', param)$. We say that $\mathcal{M}$ is *δ-partially resamplable* if there is a PT algorithm Rsmp such that $\mathsf{Rsmp}(1^k, I, \mathbf{m}^*, \mathbf{a}^*, param)$ is identically distributed as $\mathsf{Resamp}_{\mathcal{M},\delta}(1^k, I, \mathbf{m}^*, \mathbf{a}^*, param)$. Such an algorithm Rsmp is called a *δ-partial resampling algorithm* of $\mathcal{M}$. If a message sampler is already fully resamplable then it's δ-partially resamplable for any PT function δ.

3 Selective-Opening Security for D-PKE

3.1 Security Notions

Bellare, Dowsley, and Keelveedhi [4] were the first to consider selective-opening security of deterministic PKE (D-PKE). They propose a "simulation-based" semantic security notion, but then show that this definition is unachievable in both the standard model and the non-programmable random-oracle model (NPROM), even if the messages are uniform and independent. To address this, we introduce an alternative, "comparison-based" semantic-security notion that generalizes the original PRIV definition for D-PKE of Bellare, Boldyreva, and O'Neill [2]. In particular, our notion follows the IND-SO-CPA notion of Bellare, Hofheinz, and Yilek (BHY) [9] in the sense that we compare what partial information the adversary learns from the unopened messages, versus messages resampled from the same conditional distribution.

D-SO-CPA1 SECURITY. Let $\mathsf{PKE} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ be a D-PKE scheme. To a message sampler $\mathcal{M}$ and an adversary $A = (A.\text{pg}, A.\text{cor}, A.\text{g}, A.\text{f})$, we associate the experiment in Figure 1 for every $k \in \mathbb{N}$. We say that DE is D-SO-CPA1 secure for a class $\mathscr{M}$ of resamplable message samplers and a class $\mathscr{A}$ of adversaries if for every $\mathcal{M} \in \mathscr{M}$ and any $A \in \mathscr{A}$,

$$\begin{aligned} &\mathbf{Adv}^{\text{d-so-cpa1}}_{\mathsf{DE},A,\mathcal{M}}(\cdot) \\ = \;& \Pr\left[\text{D-CPA1-REAL}^{A,\mathcal{M}}_{\mathsf{DE}}(\cdot) \Rightarrow 1\right] - \Pr\left[\text{D-CPA1-IDEAL}^{A,\mathcal{M}}_{\mathsf{DE}}(\cdot) \Rightarrow 1\right] \end{aligned}$$

is negligible. In these games, the adversary $A.\text{pg}$ first creates some parameter $param$ to feed the message sampler $\mathcal{M}$. Note that $A.\text{pg}$ is not given the public key, and thus messages $\mathbf{m}_1$ created by $\mathcal{M}$ are independent of the public key, a necessary restriction of D-PKE pointed out by Bellare et al. [2]. Next, adversary $A.\text{cor}$ will be given both the public key and the ciphertexts $\mathbf{c}$, and decides which set I of indices that it'd like to open $\mathbf{c}[I]$. It then passes its state to adversary $A.\text{g}$. The latter is also given $(\mathbf{m}_1[I], \mathbf{a}[I])$ and has to output some partial information ω of the message vector $\mathbf{m}_1$.
Game D-CPA1-REAL$^{A,\mathcal{M}}_{\mathsf{DE}}$ returns 1 if the string ω above matches the output of $A.\text{f}(\mathbf{m}_1, param)$ which is the partial information of interest to the adversary. On the other hand, game D-CPA1-IDEAL$^{A,\mathcal{M}}_{\mathsf{DE}}$ returns 1 if ω is matches the output of $A.\text{f}(\mathbf{m}_0, param)$, where $\mathbf{m}_0$ is the resampled message vector by $\mathsf{Resamp}_{\mathcal{M}}(1^k, \mathbf{m}_1[I], \mathbf{a}[I], I, param)$. Note that in both games, $A.\text{f}$ is not given the public key pk, otherwise it can encrypt the messages it receives and output the resulting ciphertexts, while $A.\text{g}$ outputs $\mathbf{c}$. Again, this issue is pointed out in [2]: since encryption is deterministic, the ciphertexts themselves are some partial information about the messages. D-PKE can only hope to protect partial information of $\mathbf{m}$ that is independent of pk, and $A.\text{f}$ is therefore stripped of access to pk.

DISCUSSION. For selective-opening attacks against a D-PKE scheme in which an adversary can open d messages, it is clear that the message sampler must be (μ, d)-entropic, where $2^{-\mu(\cdot)}$ is a negligible function, for any meaningful privacy to be achievable. For convenience of discussion, let's say that a scheme is D-SO-CPA1[d] secure if it's D-SO-CPA1 secure for all (μ, d)-entropic, fully resamplable message samplers and all PT adversaries that open at most d ciphertexts, for any μ such that $2^{-\mu(\cdot)}$ is a negligible function. (The resamplability restriction is dropped for $d = 0$.) The D-SO-CPA1[0] security corresponds to the PRIV notion of Bellare et al. [2].[5]

We note that it is unclear if D-SO-CPA1[∞] security implies the classic PRIV security: the latter doesn't allow opening, but it can handle a broader class of message samplers. Our goal is to find D-PKE schemes that offer D-SO-CPA1[d] security for any value of d, including the important special cases $d = 0$ (PRIV security) and $d = \infty$ (unbounded opening).

SEPARATION. In Appendix A, we show that the standard PRIV notion of D-PKE doesn't imply D-SO-CPA1. Our construction relies on the recent result of Hofheinz, Rao, and Wichs [17] that separates the standard IND-CPA notion and IND-SO-CPA of randomized PKE. Specifically, we build a contrived D-PKE scheme that is

[5] A technical difference is that, to be consistent with [4], we require the "partial information" to be an efficiently computable function of the messages. This formulation can be shown equivalent to a definition in the style of [2] up to a difference of one in the size of the message vectors output by $\mathcal{M}$, following [6, Appendix A].

```
Game D-CPA1-REAL$_{\mathsf{DE}}^{A,\mathcal{M}}(k)$
$param \leftarrow_{\$} A.\mathrm{pg}(1^k)$
$(pk, sk) \leftarrow_{\$} \mathsf{Kg}(1^k)$
$(\mathbf{m}_1, \mathbf{a}) \leftarrow_{\$} \mathcal{M}(1^k, param)$
For $i = 1$ to $|\mathbf{m}|$ do
    $\mathbf{c}[i] \leftarrow \mathsf{Enc}(pk, \mathbf{m}_1[i])$
$(state, I) \leftarrow_{\$} A.\mathrm{cor}(pk, \mathbf{c}, param)$
$\omega \leftarrow_{\$} A.\mathrm{g}(state, \mathbf{m}_1[I], \mathbf{a}[I])$
Return $(\omega = A.\mathrm{f}(\mathbf{m}_1, param))$
```

```
Game D-CPA1-IDEAL$_{\mathsf{DE}}^{A,\mathcal{M}}(k)$
$param \leftarrow_{\$} A.\mathrm{pg}(1^k)$ ; $(pk, sk) \leftarrow_{\$} \mathsf{Kg}(1^k)$
$(\mathbf{m}_1, \mathbf{a}) \leftarrow_{\$} \mathcal{M}(1^k, param)$
For $i = 1$ to $|\mathbf{m}|$ do
    $\mathbf{c}[i] \leftarrow \mathsf{Enc}(pk, \mathbf{m}_1[i])$
$(state, I) \leftarrow_{\$} A.\mathrm{cor}(pk, \mathbf{c}, param)$
$\mathbf{m}_0 \leftarrow_{\$} \mathsf{Resamp}_{\mathcal{M}}(1^k, \mathbf{m}_1[I], \mathbf{a}[I], I, param)$
$\omega \leftarrow_{\$} A.\mathrm{g}(state, \mathbf{m}_1[I], \mathbf{a}[I])$
Return $(\omega = A.\mathrm{f}(\mathbf{m}_0, param))$
```

Figure 1: **Games to define D-SO-CPA**1 **security.**

```
Game D-CPA2-REAL$_{\mathsf{DE}}^{A,\mathcal{M},\delta}(k)$
$param \leftarrow_{\$} A.\mathrm{pg}(1^k)$
$(pk, sk) \leftarrow_{\$} \mathsf{Kg}(1^k)$
$(\mathbf{m}, \mathbf{a}) \leftarrow_{\$} \mathcal{M}(1^k, param)$
For $i = 1$ to $|\mathbf{m}|$ do
    $\mathbf{c}[i] \leftarrow \mathsf{Enc}(pk, \mathbf{m}[i])$
$(state, I) \leftarrow_{\$} A.\mathrm{cor}(pk, \mathbf{c}, param)$
$\omega \leftarrow_{\$} A.\mathrm{g}(state, \mathbf{m}[I], \mathbf{a}[I])$
Return $(\omega = A.\mathrm{f}(\delta(\mathbf{m}), param))$
```

```
Game D-CPA2-IDEAL$_{\mathsf{DE}}^{A,\mathcal{M},\delta}(k)$
$param \leftarrow_{\$} A.\mathrm{pg}(1^k)$ ; $(pk, sk) \leftarrow_{\$} \mathsf{Kg}(1^k)$
$(\mathbf{m}, \mathbf{a}) \leftarrow_{\$} \mathcal{M}(1^k, param)$
For $i = 1$ to $|\mathbf{m}|$ do
    $\mathbf{c}[i] \leftarrow \mathsf{Enc}(pk, \mathbf{m}[i])$
$(state, I) \leftarrow_{\$} A.\mathrm{cor}(pk, \mathbf{c}, param)$
$z \leftarrow_{\$} \mathsf{Resamp}_{\mathcal{M},\delta}(1^k, \mathbf{m}[I], \mathbf{a}[I], I, param)$
$\omega \leftarrow_{\$} A.\mathrm{g}(state, \mathbf{m}[I], \mathbf{a}[I])$
Return $(\omega = A.\mathrm{f}(z, param))$
```

Figure 2: **Games to define D-SO-CPA**2 **security.**

PRIV-secure in the standard model, but subject to the following D-SO-CPA1 attack. The message sampler picks a string $s \leftarrow_{\$} \{0,1\}^{\ell(k)}$ and then secret-share it to $v(k)$ shares $\mathbf{x}[1], \ldots, \mathbf{x}[v(k)]$ such that any $t(k)$ shares reveal no information about the secret s. Let $\mathbf{m}[i] \leftarrow \mathbf{x}[i] \,\|\, \mathbf{u}[i]$ for every $i \in \{1, \ldots, v(k)\}$, where $\mathbf{u}[i] \leftarrow_{\$} \{0,1\}^{2\ell(k)}$. Since s is uniform, any $t+1$ shares $\mathbf{x}[i]$ are uniform and independent. Thus, this message sampler is $(3\ell, t)$-entropic. We show that it is also efficiently resamplable. Surprisingly, there is an efficient SOA adversary $(A.\mathrm{cor}, A.\mathrm{g})$ that opens just t ciphertexts and can recover all strings $\mathbf{x}[i]$. Next, $A.\mathrm{g}$ outputs $\mathbf{x}[1] \oplus \cdots \oplus \mathbf{x}[v(k)]$, and $A.\mathrm{f}$ outputs the checksum of the first ℓ bits of the given messages. The adversary A thus wins with advantage $1 - 2^{\ell(k)}$.

D-SO-CPA2 SECURITY. The D-SO-CPA1 security notion only guarantees to protect messages that are fully resamplable. The D-SO-CPA2 notion strengthens that protection, requiring privacy of $\delta(\mathbf{m}, param)$ for any entropic message sampler $\mathcal{M}$ and any δ such that $\mathcal{M}$ is δ-partially resamplable. In Section 5, we'll see a concrete use of this extra protection, where (i) we have a sampler $\mathcal{M}$ that is not fully resamplable, but (ii) each message itself is a ciphertext, and there's a function δ such that the plaintexts underneath $\mathbf{m}$ are $\delta(\mathbf{m}, param)$ and $\mathcal{M}$ is δ-partially resamplable. Formally, let

$$\begin{aligned} &\mathbf{Adv}^{\text{d-so-cpa2}}_{\mathsf{DE},A,\mathcal{M},\delta}(\cdot) \\ = \;& \Pr\Big[\text{D-CPA2-REAL}^{A,\mathcal{M},\delta}_{\mathsf{DE}}(\cdot) \Rightarrow 1\Big] - \Pr\Big[\text{D-CPA2-IDEAL}^{A,\mathcal{M},\delta}_{\mathsf{DE}}(\cdot) \Rightarrow 1\Big], \end{aligned}$$

where games D-CPA2-REAL$^{A,\mathcal{M},\delta}_{\mathsf{DE}}$ and D-CPA2-IDEAL$^{A,\mathcal{M},\delta}_{\mathsf{DE}}$ are defined in Figure 2. In these games, adversary $A.\mathrm{f}$ is given either $\delta(\mathbf{m}_1)$ in the real game, or the output of $\mathsf{Resamp}_{\mathcal{M},\delta}(1^k, \mathbf{m}_1[I], \mathbf{a}[I], I, param)$ in the ideal game. We say that DE is D-SO-CPA2 secure if $\mathbf{Adv}^{\text{d-so-cpa2}}_{\mathsf{DE},A,\mathcal{M},\delta}(\cdot)$ is negligible for any (μ, d)-entropic message sampler $\mathcal{M}$ such that $2^{-\mu}$ is a negligible function, any PT adversary A that opens at most d ciphertexts, and any PT functions δ such that $\mathcal{M}$ is δ-partially resamplable.

WEAK EQUIVALENCE. Clearly, the D-SO-CPA2 notion implies D-SO-CPA1: the latter is the special case of the former for fully resamplable samplers, and for a specific function $\delta(\mathbf{m}, param)$ that simply returns $\mathbf{m}$. Below, we'll show that if we just restrict to fully resamplable samplers, the D-SO-CPA1 notion actually implies D-SO-CPA2. This is expected, because on an entropic, fully resamplable $\mathcal{M}$, both notions promise to protect any partial information of $\mathbf{m}$ that is independent of the public key.

Proposition 3.1 Let $\mathcal{M}$ be a fully resamplable sampler, and let δ be a PT function. Then for any adversary A, there is an adversary B such that

$$\mathbf{Adv}^{\text{d-so-cpa2}}_{\mathsf{DE},A,\mathcal{M},\delta}(\cdot) \le \mathbf{Adv}^{\text{d-so-cpa1}}_{\mathsf{DE},B,\mathcal{M}}(\cdot) \ .$$

$\underline{\mathsf{DE.Kg}(1^k)}$	$\underline{\mathsf{DE.Enc}(pk, m)}$	$\underline{\mathsf{DE.Dec}(sk, c)}$
$(ek, td) \leftarrow_{\$} \mathsf{LT.IKg}(1^k)$	$(hk, ek) \leftarrow pk$; $r \leftarrow H(hk \parallel 0 \parallel m, \mathsf{LT.il}(k))$	$(hk, td) \leftarrow sk$
$hk \leftarrow_{\$} \{0,1\}^k$	$trap \leftarrow \mathsf{LT.Eval}(ek, r)$	$(trap, y) \leftarrow c$
Return $((hk, ek), (hk, td))$	$y \leftarrow H(hk \parallel 1 \parallel r, \lvert m \rvert) \oplus m$	$r \leftarrow \mathsf{LT.Inv}(td, trap)$
	Return $(trap, y)$	Return $H(hk \parallel 1 \parallel r, \lvert y \rvert) \oplus y$

Figure 3: **D-PKE scheme** $\mathsf{DE1}[H, \mathsf{LT}]$**.**

The adversary B opens as many ciphertexts as A, and its running time is about that of A plus the time to run δ.

Proof: Let B be the adversary that is identical to A, but B.f behaves as follows. When it's given a vector $\mathbf{m}$ and parameter *param*, it'll run $z \leftarrow \delta(\mathbf{m}, param)$ and then outputs $A.\mathrm{f}(z, param)$. Then $\mathbf{Adv}^{\text{d-so-cpa1}}_{\mathsf{DE},B,\mathcal{M}}(\cdot) = \mathbf{Adv}^{\text{d-so-cpa2}}_{\mathsf{DE},A,\mathcal{M},\delta}(\cdot)$. ▮

In the remainder of the paper, we'll have 6 other notions. Any notion xxx considers an arbitrary message sampler $\mathcal{M}$ with a function δ such that $\mathcal{M}$ is δ-partially resamplable. One can consider a variant xxx1 of xxx, in which the message sampler is fully resamplable and only the specific function $\delta(\mathbf{m}, param) = \mathbf{m}$ is considered, and then establish a weak equivalence between xxx1 and xxx. However, it will lead to a proliferation of 12 definitions. We therefore choose to present just the stronger notion xxx.

CCA EXTENSION. To add a CCA flavor to D-SO-CPA2, a notion which we call D-SO-CCA, one would allow adversaries A.cor and A.g oracle access to $\mathsf{Dec}(sk, \cdot)$ with the restriction that they are forbidden from querying a ciphertext in the given $\mathbf{c}$ to this oracle. Let D-CCA-REAL and D-CCA-IDEAL be the corresponding experiments, and define

$$\begin{aligned}&\mathbf{Adv}^{\text{d-so-cca}}_{\mathsf{DE},A,\mathcal{M},\delta}(\cdot)\\ = \;&\Pr\left[\text{D-CCA-REAL}^{A,\mathcal{M},\delta}_{\mathsf{DE}}(\cdot) \Rightarrow 1\right] - \Pr\left[\text{D-CCA-IDEAL}^{A,\mathcal{M},\delta}_{\mathsf{DE}}(\cdot) \Rightarrow 1\right] .\end{aligned}$$

We say that DE is D-SO-CCA secure if $\mathbf{Adv}^{\text{d-so-cca}}_{\mathsf{DE},A,\mathcal{M},\delta}(\cdot)$ is negligible for any (μ, d)-entropic message sampler $\mathcal{M}$ such that $2^{-\mu}$ is a negligible function, any PT adversary A that opens at most d ciphertexts, and any PT functions δ such that $\mathcal{M}$ is δ-partially resamplable.

3.2 Achieving D-SO-CPA2 Security

While the simulation-based definition of Bellare et al. [4] is impossible to achieve even in the non-programmable random-oracle model (NPROM), we show that it is possible to build a D-SO-CPA2 secure scheme in the NPROM. A close variant of our scheme is shown to be PRIV-secure in the standard model [7]. Our scheme can handle messages of any length, and is highly efficient: the asymmetric cost is fixed and thus the amortized cost is about as cheap as a symmetric encryption. It's also highly practical on short messages. The only public-key primitive that it uses is a lossy trapdoor function [23], which has practical instantiations, e.g., both Rabin and RSA are lossy [21, 24].

ACHIEVING D-SO-CPA2 SECURITY. To handle arbitrary-length messages, we use a hash function H of arbitrary input and output length. On input $(x, \ell) \in \{0,1\}^* \times \mathbb{N}$, the hash returns $y = H(x, \ell) \in \{0,1\}^\ell$. Our scheme $\mathsf{DE1}[H, \mathsf{LT}]$ is shown in Figure 3, where LT is a lossy trapdoor function with domain $\{0,1\}^{\mathsf{LT.il}}$. Theorem 3.2 below shows that $\mathsf{DE1}$ is D-SO-CPA2 secure in the NPROM. The proof is in Appendix B.1. We stress that for (μ, ∞)-entropic message samplers, our scheme allows the adversary to open as many ciphertexts as it wishes.

Theorem 3.2 Let LT be a lossy trapdoor function with lossiness τ. Let $\mathcal{M}$ be a (μ, d)-entropic message sampler, and let δ be a function such that $\mathcal{M}$ is δ-partially resamplable. Let $\mathsf{DE1}[H, \mathsf{LT}]$ be as above. In the NPROM, for any adversary A opening at most d ciphertexts, there is an adversary D such that

$$\mathbf{Adv}^{\text{d-so-cpa2}}_{\mathsf{DE1}[H,\mathsf{LT}],A,\mathcal{M},\delta}(k) \le \frac{4q(k)}{2^k} + \frac{4q(k)v(k)}{2^{\mu(k)}} + \frac{v(k)(v(k) + 4q(k))}{2^{\tau(k)}} + 2\mathbf{Adv}^{\text{ltdf}}_{\mathsf{LT},D}(k),$$

where $q(k)$ is the total number of random-oracle queries of A and $\mathcal{M}$, and $v(k)$ is the number of messages that $\mathcal{M}$ produces. The running time of D is about that of A plus the time to run δ and an efficient δ-partial resampling algorithm of $\mathcal{M}$ plus the time to run $\mathsf{DE1}[H, \mathsf{LT}]$ to encrypt $\mathcal{M}$'s messages. Adversary D makes at most q random-oracle queries.

Proof ideas. Let $\mathrm{RO}_1, \mathrm{RO}_2, \mathrm{RO}_3$, and RO_4 denote the oracle interface of $(A.\mathrm{pg}, \mathcal{M}), A.\mathrm{cor}, A.\mathrm{g}$, and $A.\mathrm{f}$ respectively. Initially, each interface simply calls RO. In game-based proofs of ROM-based D-PKE constructions, one often considers the event that $A.\mathrm{pg}$ or $\mathcal{M}$ queries $(hk \,\|\, x, \ell)$ to RO_1, and then let the interface lies, instead of calling $\mathrm{RO}(hk \,\|\, x, \ell)$. This allows the coins $\mathbf{r}[i] \leftarrow \mathrm{RO}(hk \,\|\, 0 \,\|\, \mathbf{m}[i], \mathsf{LT.il}(k))$ to be independent of the messages $\mathbf{m}$. The discrepancy due to the lying is tiny, since the chance that $A.\mathrm{pg}$ or $\mathcal{M}$ can make such a query is at most $q(k)/2^k$. However, in the SOA setting, this strategy creates the following subtlety. For the resampling algorithm to behave correctly, one has to give it access to RO_1. Yet the adversary $A.\mathrm{cor}$ can embed some information of hk in I, and therefore it's well possible that the resampling algorithm queries $\mathrm{RO}_1(hk \,\|\, \cdot, \cdot)$. This issue is unique to SOA security of D-PKE: prior papers of SOA security for randomized PKE never have to deal with this. While getting around the subtlety above is not too difficult, it shows that a rigorous proof for Theorem 3.2 is not as simple as one might expect.

Suppose that $A.\mathrm{pg}$ and $\mathcal{M}$ never query $\mathrm{RO}_1(hk \,\|\, \cdot, \cdot)$. The first step in the proof is to move from an injective key ek of LT to a lossy key lk. Next, recall that the adversary $A.\mathrm{cor}$ is given $\mathsf{LT.Eval}(lk, \mathbf{r}[i])$. Since each synthetic coin $\mathbf{r}[i]$ is uniformly random and LT has lossiness τ, in the view of $A.\mathrm{cor}$, each $\mathbf{r}[i]$ has min-entropy at least $\tau(k)$. Suppose that $A.\mathrm{cor}$ doesn't make any query in $\{hk \,\|\, 0 \,\|\, \mathbf{m}[i], hk \,\|\, 1 \,\|\, \mathbf{r}[i] \mid 1 \le i \le |\mathbf{m}|\}$; this happens with probability at least $1 - q(k)v(k)/2^{\mu(k)} - q(k)v(k)/2^{\tau(k)}$. Then $A.\mathrm{cor}$ knows nothing about $\mathbf{m}$, and thus I is conditionally independent of $\mathbf{m}$, given *param*. Hence in the view of $A.\mathrm{g}$, each $\mathbf{m}[i]$ (for $i \notin I$) still has min-entropy μ, and thus the chance that $A.\mathrm{g}$ can make a query in $\{hk \,\|\, 0 \,\|\, \mathbf{m}[i] \mid i \notin I\}$ is at most $v(k)q(k)/2^{\mu(k)}$.

The core of the proof is to bound the probability that the adversary $A.\mathrm{g}$ can make a query in $\{hk \,\|\, 1 \,\|\, \mathbf{r}[i] \mid i \notin I\}$. Let X_i be the random variable for the number of pre-images of $\mathsf{LT.Eval}(lk, \mathbf{r}[i])$. Although in the view of $A.\mathrm{cor}$, the average conditional min-entropy of each $\mathbf{r}[i]$ is $\tau(k)$, the same claim may *not* hold in the view of $A.\mathrm{g}$. For example, the adversary $A.\mathrm{cor}$ may choose to open all but the ciphertext of $\mathbf{m}[j]$, where j is chosen so that $X_j = \min\{X_1, \ldots, X_{v(k)}\}$: while $\mathbf{E}(1/X_i) \le 2^{-\tau(k)}$ for each fixed $i \in \{1, \ldots, v(k)\}$, the same bound doesn't work for $\mathbf{E}(1/\min\{X_1, \ldots, X_{v(k)}\})$. To get around this, note that the chance that $A.\mathrm{g}$ can make a query in $\{hk \,\|\, 1 \,\|\, \mathbf{r}[i] \mid i \notin I\}$ is at most

$$q(k) \cdot \mathbf{E}\Big(\sum_{i \notin I} \frac{1}{X_i}\Big) \le q(k) \cdot \mathbf{E}\Big(\sum_{i=1}^{v(k)} \frac{1}{X_i}\Big) \le q(k) \cdot \sum_{i=1}^{v(k)} \mathbf{E}\Big(\frac{1}{X_i}\Big) \le \frac{q(k)v(k)}{2^{\tau(k)}} \quad .$$

Finally, if I is conditionally independent of $\mathbf{m}$ given *param*, then the re-sampled string z is identically distributed as $\delta(\mathbf{m}, \mathit{param})$, even conditioning on hk, I, and *param*.[6] Hence $A.\mathrm{f}$ can query $\mathrm{RO}_4(hk \,\|\, \cdot, \cdot)$ with probability at most $q(k)/2^k$. If all bad events above don't happen then (i) in the joint view of $A.\mathrm{g}$ and $A.\mathrm{f}$, the strings $\delta(\mathbf{m}, \mathit{param})$ and z are identically distributed, and (ii) the output of $A.\mathrm{f}$ will be conditionally independent of the ciphertexts and the public key, given *param*. This means the d-so-cpa2 advantage of A is 0.

3.3 Achieving D-SO-CCA Security

To achieve D-SO-CCA security, we modify $\mathsf{DE1}$ construction as follows: In the decryption, once we recover the message m, we'll re-encrypt it and return $\perp$ if the resulting ciphertext doesn't match the given one, or the hash image of the message doesn't match the string obtained via inverting the trapdoor function. The resulting construction $\mathsf{DE2}$ is shown in Figure 4. The scheme $\mathsf{DE} = \mathsf{DE2}[H, \mathsf{LT}]$ is *unique-ciphertext*, as formalized by Bellare and Hoang [7]: for every $k \in \mathbb{N}$, every $(pk, sk) \in [\mathsf{DE.Kg}(1^k)]$, and every $m \in \{0,1\}^*$, there is at most a string c such that $\mathsf{DE.Dec}(sk, c) = m$. Theorem 3.3 below shows that $\mathsf{DE2}$ is D-SO-CCA secure in the NPROM. The re-encrypting trick for lifting CPA to CCA security in the random-oracle model dates back to a paper of Fujisaki and Okamoto [15], but that work only considers randomized PKE and there's no opening. Still, the proof ideas are quite similar.

Theorem 3.3 Let LT be a lossy trapdoor function with lossiness τ. Let $\mathcal{M}$ be a (μ, d)-entropic message sampler and let δ be a function such that $\mathcal{M}$ is δ-partially resamplable. Let $\mathsf{DE2}[H, \mathsf{LT}]$ be as above. In the NPROM, for

[6]Even for the simple case that $\mathcal{M}$ is fully resamplable and outputs empty auxiliary information, and $\delta(\mathbf{m}, \mathit{param}) = \mathbf{m}$, note that if I is correlated to $\mathbf{m}$ then $\mathbf{m}$ and the re-sampled $\mathbf{m}'$ may have completely different distributions. For example, consider $\mathcal{M}$ that outputs (m_1, m_2), with $m_1 \leftarrow\$ \{00, 01\}$ and $m_2 \leftarrow\$ \{10, 11\}$. Since m_1 and m_2 are independent, $\mathcal{M}$ is fully resamplable. Let $I = \{1\}$ if $m_1 = 00$, and $I = \{2\}$ otherwise. Then $\Pr[\mathbf{m}' = (00, 11)] = 3/8$, whereas $\Pr[\mathbf{m} = (00, 11)] = 1/4$.

```
$\mathsf{DE.Kg}(1^k)$
$(ek, td) \leftarrow\!\!\$\ \mathsf{LT.IKg}(1^k)$
$hk \leftarrow\!\!\$\ \{0,1\}^k$
$pk \leftarrow (hk, ek)$
$sk \leftarrow (hk, ek, td)$
Return $(pk, sk)$
```

```
$\mathsf{DE.Enc}(pk, m)$
$(hk, ek) \leftarrow pk$
$r \leftarrow H(hk \,\|\, 0 \,\|\, m, \mathsf{LT.il}(k))$
$trap \leftarrow \mathsf{LT.Eval}(ek, r)$
$y \leftarrow H(hk \,\|\, 1 \,\|\, r, |m|) \oplus m$
Return $(trap, y)$
```

```
$\mathsf{DE.Dec}(sk, c)$
$(hk, ek, td) \leftarrow sk$ ; $(trap, y) \leftarrow c$
$r \leftarrow \mathsf{LT.Inv}(td, trap)$ ; $trap' \leftarrow \mathsf{LT.Eval}(ek, r)$
$m \leftarrow H(hk \,\|\, 1 \,\|\, r, |y|) \oplus y$
$r' \leftarrow H(hk \,\|\, 0 \,\|\, m, \mathsf{LT.il}(k))$
If $r' \neq r$ or $trap' \neq trap$ then return $\perp$
Return $m$
```

Figure 4: **D-PKE scheme** $\mathsf{DE} = \mathsf{DE2}[H, \mathsf{LT}]$. If LT is a lossy trapdoor permutation then in the decryption algorithm, the computation of $trap'$ and the check $trap' \neq trap$ can be omitted.

any adversary A opening at most d ciphertexts, there is an adversary D such that

$$\begin{aligned}\mathbf{Adv}^{\text{d-so-cca}}_{\mathsf{DE2}[H,\mathsf{LT}],A,\mathcal{M},\delta}(k) \quad &\leq \quad \frac{2p(k)}{2^{\mathsf{LT.il}(k)}} + \frac{10q(k)}{2^k} + \frac{4q(k)v(k)}{2^{\mu(k)}} \\ &\quad + \frac{v(k)(v(k)+8q(k))}{2^{\tau(k)}} + 2\mathbf{Adv}^{\text{ltdf}}_{\mathsf{LT},D}(k).\end{aligned}$$

where $p(k)$ is the number of decryption-oracle queries of A, $q(k)$ is the total number of random-oracle queries of A and $\mathcal{M}$, and $v(k)$ is the number of messages that $\mathcal{M}$ produces. The running time of D is about that of A plus the time to run δ and an efficient δ-partial resampling algorithm of $\mathcal{M}$, plus the time to run $\mathsf{DE2}[H, \mathsf{LT}]$ to encrypt $\mathcal{M}$'s messages and decrypt A's decryption queries. Adversary D makes at most $2q$ random-oracle queries.

Proof: Let Rsmp be an efficient δ-partial resampling algorithm for $\mathcal{M}$. Consider games G_1 and G_2 in Figure 5. Then

$$\mathbf{Adv}^{\text{d-so-cca}}_{\mathsf{DE2}[H,\mathsf{LT}],A,\mathcal{M}}(\cdot) = 2\Pr[G_1(\cdot) \Rightarrow 1] - 1 \ .$$

Game G_2 is identical to game G_1, except for the following. In procedure $\text{Dec}(c)$, instead of using the decryption of $\mathsf{DE2}$ to decrypt c, we maintain the set Dom of the suffixes of random-oracle queries (x, ℓ) that A.cor and A.g make such that $x[1, k+1] = hk \,\|\, 0$ and $\ell = \mathsf{LT.il}(k)$. If there's $m \in \mathsf{Dom}$ such that the corresponding ciphertext of m is c then we return m; otherwise return $\perp$. Wlog, assume that A.cor stores all random-oracle queries/answers in its state; that is, both A.cor and A.g also can track Dom and implement the Dec procedure of game G_2 on their own, without calling the decryption oracle.[7] Let $\mathsf{Range} = \{\mathsf{DE2.Enc}(pk, m) \mid m \in \mathsf{Dom}\}$. On a query $c \in \mathsf{Range}$, the procedures Dec of both games have the same behavior, due to the correctness of the decryption of $\mathsf{DE2}$. Wlog, assume that both A.cor and A.g never query $c \in \mathsf{Range}$ to the decryption oracle. (Adversaries A.cor and A.g are thus assumed to maintain the corresponding ciphertexts of messages in Dom. But this needs additional queries to the random oracle, so the total random-oracle queries of these two adversaries is now at most $2q$.)

Assume that A.pg and $\mathcal{M}$ never make a random-oracle query (x, ℓ) such that the k-bit suffix of x is hk. This happens with probability at least $1 - q(k)/2^k$. The adversaries can distinguish the games if and only if they can trigger Dec of game G_1 to produce non-$\perp$ output. Let $c = (trap, y)$ be a decryption-oracle query. Let $r = \mathsf{LT.Inv}(td, trap)$ and $m = \text{RO}(hk \,\|\, 1 \,\|\, r) \oplus y$. Due to the unique-ciphertext property of $\mathsf{DE2}$, if this can trigger the Dec procedure of game G_1 to return a non-$\perp$ answer, we must have $m \notin \{\mathbf{m}[1], \ldots, \mathbf{m}[|\mathbf{m}|]\} \cup \mathsf{Dom}$. Then there is no prior random-oracle query $(x, \mathsf{LT.il}(k))$ such that $x = hk \,\|\, 0 \,\|\, m$. Hence procedure Dec of game G_1 will return a non-$\perp$ answer only if $r = \text{RO}(hk \,\|\, 0 \,\|\, m, \mathsf{LT.il}(k))$, which happens with probability $2^{-\mathsf{LT.il}(k)}$. Multiplying for $p(k)$ decryption-oracle queries,

$$\Pr[G_1(k) \Rightarrow 1] - \Pr[G_2(k) \Rightarrow 1] \leq q(k)/2^k + p(k)/2^{\mathsf{LT.il}(k)} \ .$$

Now in game G_2, the decryption oracle always return $\perp$, and thus wlog, assume that the adversaries never make a decryption query, meaning that they only launch a D-SO-CPA2 attack. Hence

$$2\Pr[G_2(\cdot) \Rightarrow 1] = \mathbf{Adv}^{\text{d-so-cpa2}}_{\mathsf{DE2}[H,\mathsf{LT}],A,\mathcal{M}}(\cdot) \ .$$

[7] This assumption crucially relies on our use of a domain separation in hashing the coins r and the messages m: we employ $H(hk \,\|\, 0 \,\|\, \cdot, \cdot)$ for m, but $H(hk \,\|\, 1 \,\|\, \cdot, \cdot)$ for r. In contrast, BH's variant [7] doesn't use domain separation, and one can't make this assumption anymore: building the corresponding ciphertexts may create additional queries to $H(hk \,\|\, 0 \,\|\, \cdot, \mathsf{LT.il}(k))$, leading to a possible exponential blowup on the number of random-oracle queries.

```
Games $G_1(k)$, $G_2(k)$
    $param \leftarrow\!\!\$\ A.\mathrm{pg}^{\mathrm{RO}}(1^k)$ ; $(\mathbf{m}, \mathbf{a}) \leftarrow\!\!\$\ \mathcal{M}^{\mathrm{RO}}(1^k, param)$ ; $z_1 \leftarrow \delta(\mathbf{m}, param)$
    $hk \leftarrow\!\!\$\ \{0,1\}^k$ ; $(ek, td) \leftarrow\!\!\$\ \mathsf{LT.IKg}(1^k)$
    For $i = 1$ to $|\mathbf{m}|$ do
        $\mathbf{r}[i] \leftarrow \mathrm{RO}(hk \,\|\, 0 \,\|\, \mathbf{m}[i], \mathsf{LT.il}(k))$ ; $trap \leftarrow \mathsf{LT.Eval}(ek, \mathbf{r}[i])$
        $y \leftarrow \mathrm{RO}_1(hk \,\|\, 1 \,\|\, \mathbf{r}[i], |\mathbf{m}[i]|) \oplus \mathbf{m}[i]$ ; $\mathbf{c}[i] \leftarrow (trap, y)$
    $\mathsf{Dom} \leftarrow \emptyset$ ; $(state, I) \leftarrow\!\!\$\ A.\mathrm{cor}^{\mathrm{Dec},\mathrm{ROSim}}((hk, ek), \mathbf{c}, param)$
    $\omega \leftarrow\!\!\$\ A.\mathrm{g}^{\mathrm{Dec},\mathrm{ROSim}}(state, \mathbf{m}[I], \mathbf{a}[I])$
    $z_0 \leftarrow\!\!\$\ \mathsf{Rsmp}^{\mathrm{RO}}(1^k, \mathbf{m}[I], \mathbf{a}[I], I, param)$ ; $b \leftarrow\!\!\$\ \{0,1\}$ ; $t \leftarrow\!\!\$\ A.\mathrm{f}^{\mathrm{RO}}(z_b, param)$
    If $(\omega = t)$ then return $b$ else return $1 - b$

Procedure ROSim$(x, \ell)$
    If $|x| > k+1$ and $x[1, k+1] = hk \,\|\, 0$ then $\mathsf{Dom} \leftarrow \mathsf{Dom} \cup \{x[k+2, |x|]\}$
    Return $\mathrm{RO}(x, \ell)$
```

```
Procedure Dec$(c)$    // of game $G_1$
    $sk \leftarrow (hk, td)$
    $m \leftarrow \mathsf{DE1}[H, \mathsf{LT}].\mathsf{Dec}(sk, c)$
    Return $m$
```

```
Procedure Dec$(c)$    // of game $G_2$
    For $m \in \mathsf{Dom}$ do
        If $c = \mathsf{DE1}[H, \mathsf{LT}].\mathsf{Enc}(pk, m)$ then
            Return $m$
    Return $\perp$
```

Figure 5: **Games G_1 and G_2 of the proof of Theorem 3.3.** Their procedures Dec are in the bottom-left and bottom-right panels, respectively.

But DE2 and DE1 only differ in the decryption algorithms, which doesn't affect the D-SO-CPA security. Hence from Theorem 3.2, we can construct a distinguisher D of the claimed running time such that

$$\begin{aligned}\mathbf{Adv}^{\text{d-so-cpa2}}_{\mathsf{DE2}[H,\mathsf{LT}],A,\mathcal{M},\delta}(k) \quad &\le \quad \frac{8q(k)}{2^k} + \frac{8q(k)v(k)}{2^{\mu(k)}} + \frac{v(k)(v(k)+8q(k))}{2^{\tau(k)}} \\ &\quad + 2\mathbf{Adv}^{\mathrm{ltdf}}_{\mathsf{LT},D}(k) \ .\end{aligned}$$

(Note that the bound above is for adversaries who make at most $2q$ random-oracle queries.) Summing up,

$$\begin{aligned}\mathbf{Adv}^{\text{d-so-cca}}_{\mathsf{DE2}[H,\mathsf{LT}],A,\mathcal{M},\delta}(k) \quad &\le \quad \frac{2p(k)}{2^{\mathsf{LT.il}(k)}} + \frac{10q(k)}{2^k} + \frac{4q(k)v(k)}{2^{\mu(k)}} \\ &\quad + \frac{v(k)(v(k)+8q(k))}{2^{\tau(k)}} + 2\mathbf{Adv}^{\mathrm{ltdf}}_{\mathsf{LT},D}(k) \ .\end{aligned}$$

▮

4 Selective-Opening Security for Nonce-based PKE

Recall that D-PKE protects only unpredictable messages, but in practice, messages often have very limited entropy [14]. Hedge PKE tries to improve this situation by adding the unpredictability of coins. However, the coins generated by Dual EC are completely determined by Big Brother, and those by the buggy Debian RNG have only about 15 bits of min-entropy. In a recent work, Bellare and Tackmann (BT) [11] propose the notion of nonce-based PKE to address this limitation, supporting arbitrary messages. In this section, we extend the notion of nonce-based PKE for SOA setting, and then show how to achieve this.

4.1 Security Notions

Nonce generators. A *nonce generator* NG with nonce space $\mathcal{N}$ is an algorithm that takes as input the unary encoding 1^k of the security parameter, a current state St, and a *nonce selector* σ. It then probabilistically produces a nonce $N \in \mathcal{N}$ together with an updated state St. That is, $(N, St) \leftarrow\!\!\$\ \mathsf{NG}(1^k, St, \sigma)$. A good nonce generator needs to satisfy the following properties: (i) nonces should never repeat, and (ii) each nonce is unpredictable,

```
Game $\mathrm{RP}^{A}_{\mathsf{NG}}(k)$                          Procedure $\textsc{Gen}(\sigma)$
$St \leftarrow \varepsilon$ ; $coll \leftarrow \mathsf{false}$            $(N, St) \leftarrow\!\!\$\ \mathsf{NG}(1^k, St, \sigma)$
$\mathsf{Dom} \leftarrow \emptyset$ ; $N \leftarrow\!\!\$\ A^{\textsc{Gen}}(1^k)$            If $N \in \mathsf{Dom}$ then $coll \leftarrow \mathsf{true}$
Return $(N \in \mathsf{Dom}) \vee coll$            $\mathsf{Dom} \leftarrow \mathsf{Dom} \cup \{N\}$ ; Return $N$
```

Figure 6: **Game to define security of a nonce generator NG.**

```
Game $\text{N-SO-CPA}^{A,\mathcal{M},\delta}_{\mathsf{NE},\mathsf{NG}}(k)$
For $j = 1, 2, \ldots$ do $\boldsymbol{xk}[j] \leftarrow\!\!\$\ \mathsf{NE.Sg}(1^k)$ ; $\boldsymbol{st}[j] \leftarrow \varepsilon$
$(pk, sk) \leftarrow\!\!\$\ \mathsf{NE.Kg}(1^k)$ ; $(J, state) \leftarrow\!\!\$\ A(1^k, pk)$
$(param, \boldsymbol{N}, U, \boldsymbol{\sigma}, state) \leftarrow\!\!\$\ A(state, \boldsymbol{xk}[J])$
$(\mathbf{m}, \mathbf{a}) \leftarrow\!\!\$\ \mathcal{M}(1^k, param)$ ; $z_1 \leftarrow \delta(\mathbf{m}, param)$
For $i = 1$ to $|\mathbf{m}|$ do
    $j \leftarrow U[i]$
    If $j \in J$ then $(N, \boldsymbol{st}[j]) \leftarrow\!\!\$\ \mathsf{NG}(1^k, \boldsymbol{st}[j], \boldsymbol{\sigma}[i])$ ; $\boldsymbol{N}[i] \leftarrow N$
    $\mathbf{c}[i] \leftarrow \mathsf{NE.Enc}(pk, \boldsymbol{xk}[j], \boldsymbol{N}[i], \mathbf{m}[i])$
$(I, state) \leftarrow\!\!\$\ A(state, \mathbf{c})$ ; $b \leftarrow\!\!\$\ \{0, 1\}$
For $i \in I$, $j = 1$ to $|\mathbf{m}|$ do
    If $U[i] = U[j]$ then $I \leftarrow I \cup \{j\}$
$z_0 \leftarrow\!\!\$\ \mathsf{Resamp}_{\mathcal{M},\delta}(1^k, \mathbf{m}[I], \mathbf{a}[I], param)$
$b' \leftarrow\!\!\$\ A(state, \mathbf{m}[I], \mathbf{a}[I], \boldsymbol{N}[I], \boldsymbol{xk}[U[I]], z_b)$ ; Return $(b = b')$
```

Figure 7: **Game defining N-SO-CPA security.**

even if all nonce selectors are adversarially chosen. Formally, let $\mathbf{Adv}^{\mathrm{rp}}_{\mathsf{NG},A}(k) = \Pr[\mathrm{RP}^{A}_{\mathsf{NG}}(k)]$, where game RP is defined in Figure 6. We say that NG is RP-secure if for any PT adversary A, $\mathbf{Adv}^{\mathrm{rp}}_{\mathsf{NG},A}(\cdot)$ is a negligible function.

Nonce-based PKE. A nonce-based PKE with nonce space $\mathcal{N}$ is a tuple $\mathsf{NE} = (\mathsf{NE.Kg}, \mathsf{NE.Sg}, \mathsf{NE.Enc}, \mathsf{NE.Dec})$. The key generator $\mathsf{NE.Kg}(1^k)$ generates a public key pk and an associated secret key sk. The seed generator $\mathsf{NE.Sg}(1^k)$ produces a sender seed xk. The encryption algorithm NE.Enc takes as input a public key pk, a sender seed xk, a nonce $N \in \mathcal{N}$, and a message m, and then *deterministically* returns a ciphertext c. The decryption algorithm $\mathsf{NE.Dec}(sk, \cdot)$ plays the same role as in traditional randomized PKE; it's not given the nonce or the sender seed.

Nonce-based PKE can be viewed as a way to harden the randomness at the sender side; the receiver is oblivious to this change. Security of nonce-based PKE should hold when either (i) the seed xk is secret and the nonces are unique, or (ii) the seed is leaked to the adversary, but the nonces are unpredictable to the adversary.[8]

Discussion. To formalize security of nonce-based PKE, BT define two notions, NBP1 and NBP2. Both notions are in the single-sender setting and use nonces generated from a nonce generator NG. The former notion considers the situation when the seed xk is secret, and there's no security requirement from NG, except the uniqueness of nonces. The latter notion considers the case when the seed xk is given to the adversary; now nonces generated from NG have to satisfy RP security.
When we bring SOA extension to nonce-based PKE below, there will be many changes. First, since there are multiple senders and only some of them can keep their seeds secret, one has to merge the SOA variants of NBP1 and NBP2 into a single definition. Next, because the adversary learns the seeds of some senders, the nonce generator NG must be RP-secure. If we let senders whose seeds are secret use unpredictable nonces from NG then our notion will fail to model the possibility that the adversary can corrupt the nonce generator. Therefore, in our notion, for senders whose seeds are secret, we'll let the adversary specify their nonces. We require the adversary to be *nonce-respecting*, meaning that the nonces of every single sender must be distinct.

N-SO-CPA. Let NE be a nonce-based PKE scheme and NG be a nonce generator of the same nonce space $\mathcal{N}$. Let $\mathcal{M}$ be a message sampler, but the generated messages don't have to be distinct or unpredictable. Let δ

[8] The definition of BT [11] requires that if the seed xk is secret then security should hold as long as the message/nonce pairs are unique. If one directly extends this to the SOA setting, there will be some pesky issue, as the adversary can detect equality within the message vectors by repeating the nonces. Here for simplicity, we only demand that nonces should be unique, which is analogous to nonce-based symmetric encryption. Nevertheless, our constructions are specific instantiations of BT construction, and thus meet their requirement.

be a function such that $\mathcal{M}$ is δ-partially resamplable. The game N-SO-CPA defining the N-SO-CPA security is specified in Figure 7.

Initially, the game picks seed $\boldsymbol{xk}[j] \leftarrow\$ \mathsf{NE.Sg}(1^k)$ and sets state $\boldsymbol{st}[j] \leftarrow \varepsilon$ for sender j, with $j = 1, 2, \ldots$. The adversary is then given the public key pk and has to specify the list J of senders that it wishes to get the seeds. It's then granted $\boldsymbol{xk}[J]$ and then has to provide some parameter *param* for generating $(\mathbf{m}, \mathbf{a}) \leftarrow\$ \mathcal{M}(1^k, \mathit{param})$, together with a vector $\boldsymbol{N}$ of nonces, a map U that specifies message $\mathbf{m}[i]$ belongs to sender $U[i]$, and a vector $\boldsymbol{\sigma}$ of nonce selectors for NG. Note that the messages $\mathbf{m}$ here can depend on the public key. We require that the adversary be *nonce-respecting*, meaning that $(\boldsymbol{N}[1], U[1]), (\boldsymbol{N}[2], U[2]), \ldots$ are distinct.

The game then iterates over $i = 1, \ldots, |\mathbf{m}|$ to encrypt each message $\mathbf{m}[i]$. If $i \in J$ then $\boldsymbol{N}[i]$ is overwritten by a nonce N generated by NG as follows. Let $j \leftarrow U[i]$. The nonce generator NG will read the current state $\boldsymbol{st}[j]$ of sender j and the nonce selector $\boldsymbol{\sigma}[i]$ for the message $\mathbf{m}[i]$, to generate a nonce N and update $\boldsymbol{st}[j]$. The adversary then is given the ciphertexts and has to output a set I to indicate which ciphertexts it wants to open. Note that opening $\mathbf{c}[i]$ returns not only $(\mathbf{m}[i], \mathbf{a}[i])$ but also the associated nonce and sender seed. Moreover, if the adversary opens a message belonging to sender j, then any other messages of this sender are considered open. Finally, the game resamples z_0, and let $z_1 \leftarrow \delta(\mathbf{m}, \mathit{param})$. It picks $b \leftarrow\$ \{0, 1\}$, and gives the adversary z_b and $(\mathbf{m}[I], \mathbf{a}[I], \boldsymbol{xk}[U[I]], \boldsymbol{N}[I])$. The adversary has to guess the challenge bit b. Define

$$\mathbf{Adv}^{\text{n-so-cpa}}_{\mathsf{NE},\mathsf{NG},A,\mathcal{M},\delta}(k) = 2\Pr[\text{N-SO-CPA}^{A,\mathcal{M},\delta}_{\mathsf{NE},\mathsf{NG}}(k)] - 1 \ .$$

We say that NE is N-SO-CPA secure, with respect to NG, if for any message sampler $\mathcal{M}$ and any PT adversary A, and any PT function δ such that $\mathcal{M}$ is δ-partially resamplable, $\mathbf{Adv}^{\text{n-so-cpa}}_{\mathsf{NE},\mathsf{NG},A,\mathcal{M},\delta}(\cdot)$ is a negligible function.

N-SO-CCA. To add a CCA flavor to N-SO-CPA, one would give the adversary oracle access to $\mathsf{Dec}(sk, \cdot)$. Once it's given the ciphertexts $\mathbf{c}$, it's not allowed to query any $\mathbf{c}[i]$ to the decryption oracle. Let N-SO-CCA be the corresponding game, and define $\mathbf{Adv}^{\text{n-so-cca}}_{\mathsf{NE},\mathsf{NG},A,\mathcal{M},\delta}(k) = 2\Pr[\text{N-SO-CCA}^{A,\mathcal{M},\delta}_{\mathsf{NE},\mathsf{NG}}(k)] - 1$.

SIMULATION-BASED SECURITY. One could also define an appropriate simulation-based notion of SOA security for nonce-based PKE, which unlike N-SO-CPA would not require the unrevealed messages to be efficiently resampleable, analogously to the SIM-SOA definition for randomized PKE in [9]. However, we conjecture that such a definition is impossible to achieve. We leave this as an open question. In any case, a simulation-based definition of SOA security for nonce-based PKE will indeed be impossible to achieve later when we lift the primitive to the hedged setting, where an existing impossibility result for a simulation-based notion of SOA security for deterministic PKE [4] applies (because hedged PKE generalizes deterministic PKE).

4.2 Separation

We now show that the standard notions for nonce-based PKE of BT [11] do not imply N-SO-CPA. Our separation is based on the recent result of Hofheinz, Rao, and Wichs (HRW) [17] to show that IND-CCA doesn't imply the notion IND-SO-CPA for randomized PKE.

HRW CONSTRUCTION. Our counterexample is based on the recent (contrived) construction $\mathsf{RE}_{\mathsf{bad}} = (\mathsf{RE}_{\mathsf{bad}}.\mathsf{Kg}, \mathsf{RE}_{\mathsf{bad}}.\mathsf{Enc}, \mathsf{RE}_{\mathsf{bad}}.\mathsf{Dec})$ of HRW. The scheme $\mathsf{RE}_{\mathsf{bad}}$ is IND-CCA secure, but is vulnerable to the following SOA attack. The message sampler $\mathcal{M}(1^k, \mathit{param})$ ignores *param*, picks a secret $s \leftarrow\$ \{0, 1\}^\ell$ and then secret-shares it to $v(k)$ messages $\mathbf{m}[1], \ldots, \mathbf{m}[v(k)]$ so that any $t(k)$ shares reveal no information of the secret s. In other words, it picks $a_0, a_1, \ldots, a_t$ uniformly from $\mathsf{GF}(2^\ell)$, the finite field of size 2^ℓ, and computes $\mathbf{m}[i] \leftarrow f(i)$ for every $i \in \{1, \ldots, v(k)\}$, where $f(x) = a_0 + a_1 x + \cdots + a_t x^t$ is the corresponding polynomial in $\mathsf{GF}(2^\ell)[X]$. Recall that any $t+1$ shares will uniquely determine the polynomial f (via polynomial interpolation), and thus any $t+1$ shares are uniformly and independently random. The auxiliary information is empty. Surprisingly, there's an efficient adversary that opens only t ciphertexts and can recover all messages. We note that HRW's counter-example is based on public-coin differing-inputs obfuscation [19], which is a very strong assumption.

RESULTS. Let H be a hash function. One can model it as a random oracle, or, for a standard-model result, a primitive that BT call *hedged extractor*. BT show that one can build a nonce-based PKE achieving their notions from an arbitrary IND-CCA secure PKE RE as follows. Given seed xk, nonce N, and message m, one uses $H((xk, N, m))$ to extract synthetic coins r, and then encrypt m via RE under coins r. Now, use the scheme $\mathsf{RE}_{\mathsf{bad}}$ above to instantiate RE, and let $\mathsf{NE}_{\mathsf{bad}}[H, \mathsf{RE}_{\mathsf{bad}}]$ be the resulting nonce-based PKE. This $\mathsf{NE}_{\mathsf{bad}}[H, \mathsf{RE}_{\mathsf{bad}}]$ achieves BT's notions.

We now break the N-SO-CPA security of $\mathsf{NE}_{\mathsf{bad}}$. The message sampler $\mathcal{M}$ is as described in HRW attack, and let A be the adversary attacking $\mathsf{RE}_{\mathsf{bad}}$ as above. Note that $\mathcal{M}$ is fully resamplable, and let $\delta(\mathbf{m}, \mathit{param}) = \mathbf{m}$.

```
$\mathsf{NE1.Kg}(1^k)$
    $(ek, td) \leftarrow_\$ \mathsf{LT.IKg}(1^k)$ ; $hk \leftarrow_\$ \{0,1\}^k$
    $pk \leftarrow (hk, ek)$ ; $sk \leftarrow (hk, td)$ ; Return $(pk, sk)$

$\mathsf{NE1.Sg}(1^k)$
    $xk \leftarrow_\$ \{0,1\}^k$ ; Return $xk$

$\mathsf{NE1.Enc}(pk, xk, N, m)$
    $(hk, ek) \leftarrow pk$
    $r \leftarrow H(hk \,\|\, 0 \,\|\, (xk, N, m), \mathsf{LT.il}(k))$
    $trap \leftarrow \mathsf{LT.Eval}(pk, r)$ ; $y \leftarrow H(hk \,\|\, 1 \,\|\, r, |m|) \oplus m$
    Return $(trap, y)$

$\mathsf{NE1.Dec}(sk, c)$
    $(hk, td) \leftarrow sk$ ; $(trap, y) \leftarrow c$
    $r \leftarrow \mathsf{LT.Inv}(sk, trap)$
    $m \leftarrow H(hk \,\|\, 1 \,\|\, r, |y|) \oplus y$
    Return $m$
```

Figure 8: **Nonce-based PKE scheme** $\mathsf{NE1}[H, \mathsf{LT}]$**.**

Consider the following adversary B attacking $\mathsf{NE}_{\mathsf{bad}}$. It specifies $J = \emptyset$, meaning that it doesn't want to get any sender seed before the opening. It then lets $\boldsymbol{N}[i] = U[i] = i$, for every i. That is, each sender has only a single message. Then, when B gets the ciphertexts $\mathbf{c}$, it runs A on those $\mathbf{c}$. Note that $\mathbf{c}$ are ciphertexts of $\mathsf{RE}_{\mathsf{bad}}$, although the coins are only pseudorandom. Still, adversary A can recover all messages by opening just t ciphertexts. When B is given the messages (real or resampled), it compares that with what A recovers. Then $\mathbf{Adv}^{\text{n-so-cpa}}_{\mathsf{NE},\mathsf{NG},B,\mathcal{M},\delta}(k) \geq 1 - 2^{-\ell(k)}$, where ℓ is the length of each message.

4.3 Achieving N-SO-CPA Security

BT's construction of nonce-based PKE is simple. To encrypt a message m under a seed xk, a nonce N, and public key pk, we hash (xk, N, m) to derive a string r, and then uses a traditional randomized PKE to encrypt m under the synthetic coins r and public key pk. Here we'll use BT's construction, but the underlying randomized PKE is a randomized counterpart of the D-PKE scheme $\mathsf{DE1}$ in Section 3.2.

Formally, let H be a hash of arbitrary input and output length, meaning that $H(x, \ell)$ returns an ℓ-bit string. Let LT be a lossy trapdoor function. Our nonce-based PKE $\mathsf{NE1}[H, \mathsf{LT}]$ is described in Figure 8; it has nonce space $\{0,1\}^*$ and message space $\{0,1\}^*$. Theorem 4.1 below shows that $\mathsf{NE1}[H, \mathsf{LT}]$ is N-SO-CPA secure in the NPROM; the proof is in Appendix B.2.

Theorem 4.1 Let LT be a lossy trapdoor function with lossiness τ. Let $\mathcal{M}$ be a message sampler and let δ be a function such that $\mathcal{M}$ is δ-partially resamplable. Let $\mathsf{NE1}[H, \mathsf{LT}]$ be as above, and let NG be a nonce generator. In the NPROM, for any adversary A, there are adversaries B and D such that

$$\begin{aligned}\mathbf{Adv}^{\text{n-so-cpa}}_{\mathsf{NE1}[H,\mathsf{LT}],\mathsf{NG},A,\mathcal{M},\delta}(k) \quad &\leq \quad 2\mathbf{Adv}^{\text{ltdf}}_{\mathsf{LT},B}(k) + 8q(k)v(k) \cdot \mathbf{Adv}^{\text{rp}}_{\mathsf{NG},D}(k) \\ &\quad + \frac{7v(k)(q(k)+v(k))}{2^k} + \frac{12v(k)(q(k)+v(k))}{2^{\tau(k)}},\end{aligned}$$

where v is the number of messages that $\mathcal{M}$ generates, and q bounds the total number of random-oracle queries that A and $\mathcal{M}$ make. The running time of B or D is about the time to run game $\text{N-SO-CPA}^{A,\mathcal{M},\delta}_{\mathsf{NE},\mathsf{NG}}$, but using an efficient δ-partial resampling algorithm of $\mathcal{M}$ instead of $\mathsf{Resamp}_{\mathcal{M},\delta}$. Each of B and D makes at most q random-oracle queries.

4.4 Achieving N-SO-CCA Security

To strengthen $\mathsf{NE1}$ with CCA capability, in the encryption, we append to the ciphertext a hash image of $r \,\|\, m$. When we decrypt a ciphertext, we'll recover both r and m, and check if the hash image of $r \,\|\, m$ matches with what's given in the ciphertext. The resulting scheme $\mathsf{NE2}[H, \mathsf{LT}]$ is shown in Figure 9. The underlying randomized PKE of $\mathsf{NE2}$ is a textbook IND-CCA construction in the ROM (but LT just needs to be an ordinary trapdoor function). Theorem 4.2 below shows that $\mathsf{NE2}[H, \mathsf{LT}]$ is N-SO-CCA secure in the NPROM; the proof is in Appendix B.3.

Theorem 4.2 Let LT be a lossy trapdoor function with lossiness τ. Let $\mathcal{M}$ be a message sampler and let δ be a function such that $\mathcal{M}$ is δ-partially resamplable. Let $\mathsf{NE2}[H, \mathsf{LT}]$ be as above, and let NG be a nonce generator.

```
NE2.Kg(1^k)
    $(ek, td) \leftarrow_\$ \mathsf{LT.lKg}(1^k)$ ; $hk \leftarrow_\$ \{0,1\}^k$
    $pk \leftarrow (hk, ek)$ ; $sk \leftarrow (hk, ek, td)$
    Return $(pk, sk)$

NE2.Sg(1^k)
    $xk \leftarrow_\$ \{0,1\}^k$ ; Return $xk$

NE2.Enc(pk, xk, N, m)
    $(hk, ek) \leftarrow pk$
    $r \leftarrow H(hk \,\|\, 00 \,\|\, (xk, N, m), \mathsf{LT.il}(k))$
    $y \leftarrow H(hk \,\|\, 01 \,\|\, r, |m|) \oplus m$
    $z \leftarrow H(hk \,\|\, 10 \,\|\, r \,\|\, m, k)$
    $trap \leftarrow \mathsf{LT.Eval}(pk, r)$
    Return $(trap, y)$

NE2.Dec(sk, c)
    $(hk, ek, td) \leftarrow sk$ ; $(trap, y, z) \leftarrow c$
    $r \leftarrow \mathsf{LT.Inv}(sk, trap)$
    $trap' \leftarrow \mathsf{LT.Eval}(pk, r)$
    $m \leftarrow H(hk \,\|\, 01 \,\|\, r, |y|) \oplus y$
    $z' \leftarrow H(hk \,\|\, 10 \,\|\, r \,\|\, m, k)$
    If $(z' \neq z) \vee (trap' \neq trap)$ then return $\perp$ Return $m$
```

Figure 9: **Nonce-based PKE scheme** $\mathsf{NE2}[H, \mathsf{LT}]$**.**

In the NPROM, for any adversary A, there are adversaries B and D such that

$$\mathbf{Adv}^{\text{n-so-cca}}_{\mathsf{NE2}[H,\mathsf{LT}],\mathsf{NG},A,\mathcal{M},\delta}(k) \leq 2\mathbf{Adv}^{\text{ltdf}}_{\mathsf{LT},B}(k) + 8v(k)Q(k) \cdot \mathbf{Adv}^{\text{rp}}_{\mathsf{NG},D}(k) + \frac{2p(k)}{2^k} + \frac{7v(k)Q(k)}{2^k} + \frac{12v(k)Q(k)}{2^{\tau(k)}},$$

where v is the number of messages that $\mathcal{M}$ generates, p is the number of A's queries to the decryption oracle, q bounds the total number of random-oracle queries that A and $\mathcal{M}$ make, and $Q = q + 2p + v$. The running time of B or D is about the time to run game N-SO-CCA$^{A,\mathcal{M},\delta}_{\mathsf{NE},\mathsf{NG}}$, but using an efficient δ-partial resampling algorithm of $\mathcal{M}$ instead of $\mathsf{Resamp}_{\mathcal{M},\delta}$. Each of B and D makes at most $q + 2p$ random-oracle queries.

5 Hedged Security for Nonce-based PKE

Recall that the security of nonce-based PKE relies on the assumption that the adversary cannot obtain the secret seeds and corrupt the nonce generator simultaneously. Still, this assumption may fail in practice, and it's desirable to retain some security guarantee when seeds and nonces are bad. We capture this via the notion HN-SO-CPA that is a variant of the notion D-SO-CPA2, adapted for the nonce-based setting. A good nonce-based PKE thus has to satisfy both N-SO-CPA and HN-SO-CPA simultaneously. We then extend this treatment to the CCA setting.

5.1 Security Notions

Unpredictable samplers. Let $\mathcal{M}$ be a message sampler. We say that $\mathcal{M}$ is (μ, d)-*unpredictable* if for any $param \in \{0,1\}^*$,

(i) For any $(\mathbf{m}, \mathbf{a}) \in [\mathcal{M}(1^k, param)]$, each $\mathbf{a}[i]$ is a tuple (a_i, xk_i, N_i), where xk_i is a seed and N_i is a nonce. Moreover, $(xk_1, N_1, \mathbf{m}[1]), (xk_2, N_2, \mathbf{m}[2]), \ldots$ must be distinct.

(ii) For any $I \subseteq \{1, \ldots, v(k)\}$ such that $|I| \leq d$, and any $i \in \{1, \ldots, v(k)\} \backslash I$, for $(\mathbf{m}, \mathbf{a}) \leftarrow_\$ \mathcal{M}(1^k, param)$ the conditional min-entropy of $(\mathbf{m}[i], xk_i, N_i)$ given $(\mathbf{m}[I], \mathbf{a}[I], param)$ is at least μ, where $v(k)$ is the number of messages that $\mathcal{M}$ produces and xk_i and N_i are the seed and nonce specified by $\mathbf{a}[i]$.

Defining unpredictable samplers allows us to model the situation when the seeds, nonces, and messages are related, and quantify security based on the combined min-entropy of each message with its nonce and seed.

HN-SO-CPA security. Let NE be a nonce-based PKE scheme, and let $\mathcal{M}$ be an unpredictable message sampler. Let δ be a function such that $\mathcal{M}$ is δ-partially resamplable. Let $A = (A.\text{pg}, A.\text{cor}, A.\text{g}, A.\text{f})$ be an adversary. Define

$$\mathbf{Adv}^{\text{hn-so-cpa}}_{\mathsf{NE},A,\mathcal{M},\delta}(\cdot) = \Pr\left[\text{HN-CPA-REAL}^{A,\mathcal{M},\delta}_{\mathsf{NE}}(\cdot) \Rightarrow 1\right] - \Pr\left[\text{HN-CPA-IDEAL}^{A,\mathcal{M},\delta}_{\mathsf{NE}}(\cdot) \Rightarrow 1\right],$$

where the games are defined in Figure 10.

```
Game HN-CPA-REAL$_{\mathsf{NE}}^{A,\mathcal{M},\delta}(k)$
$param \leftarrow_\$ A.\mathrm{pg}(1^k)$ ; $(pk, sk) \leftarrow_\$ \mathsf{Kg}(1^k)$ ; $(\mathbf{m}, \mathbf{a}) \leftarrow_\$ \mathcal{M}(1^k, param)$
For $i = 1$ to $|\mathbf{m}|$ do $(a, xk, N) \leftarrow \mathbf{a}[i]$ ; $\mathbf{c}[i] \leftarrow \mathsf{Enc}(pk, xk, N, \mathbf{m}[i])$
$(state, I) \leftarrow_\$ A.\mathrm{cor}(pk, \mathbf{c}, param)$ ; $\omega \leftarrow_\$ A.\mathrm{g}(state, \mathbf{m}[I], \mathbf{a}[I])$
$z_1 \leftarrow \delta(\mathbf{m}, param)$
Return $(\omega = A.\mathrm{f}(z_1, param))$

Game HN-CPA-IDEAL$_{\mathsf{NE}}^{A,\mathcal{M},\delta}(k)$
$param \leftarrow_\$ A.\mathrm{pg}(1^k)$ ; $(pk, sk) \leftarrow_\$ \mathsf{Kg}(1^k)$ ; $(\mathbf{m}, \mathbf{a}) \leftarrow_\$ \mathcal{M}(1^k, param)$
For $i = 1$ to $|\mathbf{m}|$ do $(a, xk, N) \leftarrow \mathbf{a}[i]$ ; $\mathbf{c}[i] \leftarrow \mathsf{Enc}(pk, xk, N, \mathbf{m}[i])$
$(state, I) \leftarrow_\$ A.\mathrm{cor}(pk, \mathbf{c}, param)$ ; $\omega \leftarrow_\$ A.\mathrm{g}(state, \mathbf{m}[I], \mathbf{a}[I])$
$z_0 \leftarrow_\$ \mathsf{Resamp}_{\mathcal{M},\delta}(1^k, \mathbf{m}[I], \mathbf{a}[I], I, param)$
Return $(\omega = A.\mathrm{f}(z_0, param))$
```

Figure 10: **Games to define HN-SO-CPA security.**

$\overline{\mathsf{NE}}.\mathsf{Kg}(1^k)$	$\overline{\mathsf{NE}}.\mathsf{Enc}(pk, xk, N, m)$	$\overline{\mathsf{NE}}.\mathsf{Dec}(sk, c)$
$(pk_{\mathrm{n}}, sk_{\mathrm{n}}) \leftarrow_\$ \mathsf{NE.Kg}(1^k)$	$(pk_{\mathrm{n}}, pk_{\mathrm{d}}) \leftarrow pk$	$(sk_{\mathrm{n}}, sk_{\mathrm{d}}) \leftarrow sk$
$(pk_{\mathrm{d}}, sk_{\mathrm{d}}) \leftarrow_\$ \mathsf{DE.Kg}(1^k)$	$m' \leftarrow (m \,\|\, xk \,\|\, N)$	$y \leftarrow \mathsf{DE.Dec}(sk_{\mathrm{d}}, c)$
$pk \leftarrow (pk_{\mathrm{n}}, pk_{\mathrm{d}})$; $sk \leftarrow (sk_{\mathrm{n}}, sk_{\mathrm{d}})$	$y \leftarrow \mathsf{NE.Enc}(pk_{\mathrm{n}}, xk, N, m')$	$m' \leftarrow \mathsf{NE.Dec}(sk_{\mathrm{n}}, y)$
Return (pk, sk)	$c \leftarrow \mathsf{DE.Enc}(pk_{\mathrm{d}}, y)$	$(m \,\|\, xk \,\|\, N) \leftarrow m'$
	Return c	Return $(m \,\|\, xk \,\|\, N)$

Figure 11: **Nonce-based PKE scheme $\overline{\mathsf{NE}} = \mathsf{NtD}[\mathsf{NE}, \mathsf{DE}]$.** It uses the same seed-generating algorithm as NE.

HN-SO-CCA SECURITY. To add a CCA flavor to HN-SO-CPA, one would give A.cor and A.g oracle access to $\mathsf{Dec}(sk, \cdot)$. They are not allowed to query any $\mathbf{c}[i]$ to the decryption oracle. Let HN-CCA-REAL and HN-CCA-IDEAL be the corresponding games, and define

$$\mathbf{Adv}^{\text{hn-so-cca}}_{\mathsf{NE},A,\mathcal{M},\delta}(\cdot)$$
$$= \Pr\left[\text{HN-CCA-REAL}_{\mathsf{NE}}^{A,\mathcal{M},\delta}(\cdot) \Rightarrow 1\right] - \Pr\left[\text{HN-CCA-IDEAL}_{\mathsf{NE}}^{A,\mathcal{M},\delta}(\cdot) \Rightarrow 1\right] .$$

SEPARATION. We now show that N-SO-CCA doesn't imply HN-SO-CPA, even if $\mathcal{M}$ picks $\mathbf{m}[i] \leftarrow_\$ \{0,1\}^k$ and $\mathbf{a}[i] = (i, i, i)$, and there's no opening. Note that $\mathcal{M}$ is fully resamplable, and consider the function δ such that $\delta(\mathbf{m}, param) = param$. Let H be a hash and LT be a lossy trapdoor function. Let $\mathsf{NE}_{\mathsf{bad}}[H, \mathsf{LT}]$ be the following variant of $\mathsf{NE2}[H, \mathsf{LT}]$. To encrypt message m under public key pk, seed xk and nonce N, instead of hashing (xk, N, m) to derive synthetic coins r, we just hash (xk, N). The proof of Theorem 4.2 can be recast to justify the N-SO-CCA security $\mathsf{NE}_{\mathsf{bad}}$. However, without even opening, one can trivially break HN-SO-CPA security of $\mathsf{NE}_{\mathsf{bad}}$ as follows. First, adversary A.pg outputs an arbitrary *param*. Next, adversary A.cor stores the ciphertexts and the public key in its state, and outputs $I = \emptyset$. Adversary A.g computes $r \leftarrow H(hk \,\|\, 00 \,\|\, (1, 1))$, parses $(trap, y, z) \leftarrow \mathbf{c}[1]$, and outputs $\mathbf{m}[1] = y \oplus H(hk \,\|\, 01 \,\|\, r, |y|)$. Finally, adversary $A.\mathrm{f}(\mathbf{m}^*, param)$ simply outputs $\mathbf{m}^*[1]$. The adversaries win with advantage $1 - 2^{-k}$.

5.2 Achieving HN-SO-CPA Security

NtD TRANSFORM. We first give a transform Nonce-then-Deterministic (NtD). Let DE be a D-SO-CPA2 secure D-PKE and NE be an N-SO-CPA secure nonce-based PKE. Then $\mathsf{NtD}[\mathsf{NE}, \mathsf{DE}]$ achieves both HN-SO-CPA and N-SO-CPA security simultaneously. The resulting nonce-based PKE $\overline{\mathsf{NE}}$ is a double encryption: it first encrypts via NE, and then uses DE to encrypt the resulting ciphertext.[9] The transform NtD is shown in Figure 11, and Theorem 5.1 below confirms that it works as claimed.

DISCUSSION. To explain why NtD works, note that using an outer D-PKE on the ciphertext of NE doesn't affect its N-SO-CPA security, and thus $\overline{\mathsf{NE}} = \mathsf{NtD}[\mathsf{NE}, \mathsf{DE}]$ inherits the N-SO-CPA security of NE. For HN-SO-CPA

[9] For simplicity, we assume that the ciphertext length of $\overline{\mathsf{NE}}$ is the plaintext length of DE. One may also consider a more generalized setting in which the ciphertext length of NE is smaller than the plaintext length of DE. In this case one needs to pad 10^* to the ciphertexts of NE before feeding them to DE.

security, there are some subtle points as follows.

First, the "messages" for DE are the ciphertexts produced by NE. Now, the D-SO-CPA2 security demands that those "messages" must have good min-entropy, but we only know that the combined min-entropy of each message with its nonce and seed is μ. We need a bound, call it $\mathsf{NE.Guess}(\mu)$, to quantify the min-entropy of the ciphertexts of NE. Therefore, let $\mathsf{NE.Guess}(\mu(k))$ be biggest number that, for any seed xk, any nonce N, any message m, and any random variable X such that the conditional min-entropy of (m, xk, N) given X is at least $\mu(k)$, and $(pk, sk) \leftarrow\$ \mathsf{NE.Kg}(1^k)$ independent of (m, xk, N, X), the conditional min-entropy of $\mathsf{NE.Enc}(pk, xk, N, m)$ given X is at least $\mathsf{NE.Guess}(\mu(k))$. We say that NE is *entropy-preserving* if for any μ such that $2^{-\mu}$ is negligible, so is $2^{-\mathsf{NE.Guess}(\mu)}$. For example, one can show that $\mathsf{NE1}[H, \mathsf{LT}].\mathsf{Guess}(\mu(k)) \ge \min\{k, \mu(k)/2\} - 1$, by modeling $\mathsf{h}_{hk}(\cdot) = H(hk \,\|\, 0 \,\|\, \cdot, \mathsf{LT.il}(k))$ as a universal hash function, and using the Generalized Leftover Hash Lemma [1, Lemma 3.4]. Hence $\mathsf{NE1}$ is entropy-preserving.

Next, we need to build an adversary B attacking DE from an adversary A that attacks $\overline{\mathsf{NE}}$. Then B.pg will run $param \leftarrow\$ A.\mathrm{pg}(1^k)$, pick $(pk, sk) \leftarrow\$ \mathsf{NE.Kg}(1^k)$, and outputs $pars = (pk, sk, param)$, asking its sampler $\overline{\mathcal{M}}$ to run $\mathcal{M}$ and encrypt the resulting messages, nonces, and seeds under pk. At some point, A.cor will asks to open some ciphertexts $\mathbf{c}[I]$ to get the corresponding $\mathbf{m}[I], \boldsymbol{xk}[I], \boldsymbol{N}[I]$, but the opened "messages" that B.g receives are $\mathsf{NE.Enc}(pk, \boldsymbol{xk}[i, \boldsymbol{N}[i], \mathbf{m}[i])$. Although B.g knows the secret key sk of NE, if we use $\mathsf{NE1}$ to instantiate NE then one can't recover $(\boldsymbol{N}[i], \boldsymbol{xk}[i])$ from just $c_i = \mathsf{NE.Enc}(pk, \boldsymbol{xk}[i], \boldsymbol{N}[i], \mathbf{m}[i])$ and sk. Thanks to our explicit modeling of the auxiliary information, adversary B does get $(\boldsymbol{xk}[i], \boldsymbol{N}[i])$ when it opens $\mathbf{c}[i]$.

Finally, one has to reason about the resamplability of the constructed sampler $\overline{\mathcal{M}}$. Had we restricted our notions to fully resamplable samplers and the function $\delta(\mathbf{m}, param) = \mathbf{m}$, we would have run into problem here. Why so? The resampling algorithm $\overline{\mathsf{Rsmp}}$ of $\overline{\mathcal{M}}$ has to generate $\mathsf{NE.Enc}(pk, \boldsymbol{xk}'[i], \boldsymbol{N}'[i], \mathbf{m}'[i])$, but it only knows pk and another algorithm Rsmp to generate $\mathbf{m}'$. That is, it's unclear how to resample the seeds $\boldsymbol{xk}'$ and nonces $\boldsymbol{N}'$. Using partial resamplability solves this issue. To justify this, suppose that we need to justify the HN-SO-CPA security of $\overline{\mathsf{NE}}$ with respect to function δ. Then, we'll find *another* function $\overline{\delta}$ such that $\mathbf{Adv}^{\text{hn-so-cpa}}_{\overline{\mathsf{NE}}, A, \mathcal{M}, \delta}(\cdot) \le \mathbf{Adv}^{\text{d-so-cpa2}}_{\mathsf{DE}, B, \overline{\mathcal{M}}, \overline{\delta}}(\cdot)$, and at the same time, $\overline{\mathcal{M}}$ is $\overline{\delta}$-partially resamplable. The function $\overline{\delta}(\mathbf{x}, pars)$ works as follows. It first parses $pars$ as $(pk, sk, param)$, runs $\mathbf{m}[i] \leftarrow \mathsf{NE.Dec}(sk, \mathbf{x}[i])$, and then outputs $\delta(\mathbf{m}, param)$.

We stress that the NtD transform works in both the standard model and the NPROM. (Of course, this assumes that there are standard-model D-SO-CPA2 secure D-PKE and N-SO-CPA secure entropy-preserving nonce-based PKE.)

Theorem 5.1 Let NE be a nonce-based PKE, and let DE be a D-PKE scheme such that the ciphertext length of the former is a plaintext length of the latter. Let $\overline{\mathsf{NE}} = \mathsf{NtD}[\mathsf{NE}, \mathsf{DE}]$.

N-SO-CPA security: For any adversary A, any message sampler $\mathcal{M}$, any function δ such that $\mathcal{M}$ is δ-partially resamplable, and any nonce generator NG, there is an adversary B such that

$$\mathbf{Adv}^{\text{n-so-cpa}}_{\overline{\mathsf{NE}}, \mathsf{NG}, A, \mathcal{M}, \delta}(\cdot) \le \mathbf{Adv}^{\text{n-so-cpa}}_{\mathsf{NE}, \mathsf{NG}, B, \mathcal{M}, \delta}(\cdot) \ .$$

The running time of B is about that of A plus the running time of $\mathsf{DE.Kg}$ plus the time to run $\mathsf{DE.Enc}$ on the messages that $\mathcal{M}$ produces.

HN-SO-CPA security: For any (μ, d)-unpredictable message sampler $\mathcal{M}$, any function δ such that $\mathcal{M}$ is δ-partially resamplable, and any adversary A, there are an adversary B that opens the same number of ciphertexts, another function $\overline{\delta}$, and another $(\mathsf{NE.Guess}(\mu), d)$-entropic, $\overline{\delta}$-partially resamplable message sampler $\overline{\mathcal{M}}$ such that

$$\mathbf{Adv}^{\text{hn-so-cpa}}_{\overline{\mathsf{NE}}, A, \mathcal{M}, \delta}(\cdot) \le \mathbf{Adv}^{\text{d-so-cpa2}}_{\mathsf{DE}, B, \overline{\mathcal{M}}, \overline{\delta}}(\cdot) \ .$$

The running time of B is about that of A plus the running time of $\overline{\mathsf{NE}}.\mathsf{Kg}$ plus the time to run $\overline{\mathsf{NE}}.\mathsf{Dec}$ on v ciphertexts, where v is the number of messages that $\mathcal{M}$ produces. The running time of $\overline{\mathcal{M}}$ is about that of $\mathcal{M}$ plus the time to run $\mathsf{NE.Enc}$ on v messages.

Proof: For the first part, consider an arbitrary adversary A. Consider the adversary B in Figure 12 attacking NE. Then game $\text{N-SO-CPA}^{B, \mathcal{M}, \delta}_{\mathsf{NE}, \mathsf{NG}}$ coincides with game $\text{N-SO-CPA}^{A, \mathcal{M}, \delta}_{\overline{\mathsf{NE}}, \mathsf{NG}}$, and thus $\mathbf{Adv}^{\text{n-so-cpa}}_{\overline{\mathsf{NE}}, \mathsf{NG}, A, \mathcal{M}, \delta}(\cdot) = \mathbf{Adv}^{\text{n-so-cpa}}_{\mathsf{NE}, \mathsf{NG}, B, \mathcal{M}, \delta}(\cdot)$.

For the second part, consider an arbitrary adversary A and a message sampler $\mathcal{M}$. Consider the following message sampler $\overline{\mathcal{M}}(1^k, pars)$. It parses $param$ as a triple $(pk_{\mathrm{n}}, sk_{\mathrm{n}}, param)$, where pk_{n} and sk_{n} are public and

```
Algorithm $B(1^k, pk)$
Return $A(1^k, pk)$

Algorithm $B(state, \mathbf{c})$
$(pk_d, sk_d) \leftarrow\$ \mathsf{DE.Kg}(1^k)$
For $i = 1$ to $|\mathbf{c}|$ do $\overline{\mathbf{c}} \leftarrow \mathsf{DE.Enc}(pk_d, \mathbf{c}[i])$
Return $A(state, \overline{\mathbf{c}})$

Algorithm $B(state, \boldsymbol{xk}^*)$
Return $A(state, \boldsymbol{xk}^*)$

Algorithm $B(state, \mathbf{m}^*, \mathbf{a}^*, \boldsymbol{N}^*, \boldsymbol{xk}^*)$
Return $A(state, \mathbf{m}^*, \mathbf{a}^*, \boldsymbol{N}^*, \boldsymbol{xk}^*)$
```

Figure 12: **N-SO-CPA adversary B in the proof of Theorem 5.1.**

```
Algorithm $B.\mathrm{pg}(1^k)$
$param \leftarrow\$ A.\mathrm{pg}(1^k)$
$(pk_n, sk_n) \leftarrow\$ \mathsf{NE.Kg}(1^k)$
$pars \leftarrow (pk_n, sk_n, param)$
Return $pars$

Algorithm $B.\mathrm{cor}(pk_d, \mathbf{c}, pars)$
$(pk_n, sk_n, param) \leftarrow pars$
$pk \leftarrow (pk_d, pk_n)$
$(state, I) \leftarrow\$ A.\mathrm{cor}(pk, \mathbf{c}, param)$
$t \leftarrow (state, pars)$ ; Return $(t, I)$

Algorithm $B.\mathrm{g}(t, \mathbf{y}^*, \mathbf{a}^*)$
$(state, pars) \leftarrow t$
$(pk_n, sk_n, param) \leftarrow pars$
For $i \leftarrow 1$ to $|\mathbf{y}^*|$ do
    $\mathbf{m}[i] \leftarrow \mathsf{NE.Dec}(sk_n, \mathbf{y}[i])$
$\omega \leftarrow\$ A.\mathrm{g}(state, \mathbf{m}^*, \mathbf{a}^*)$
Return $\omega$

Algorithm $B.\mathrm{f}(z, pars)$
$(pk_n, sk_n, param) \leftarrow pars$
$t \leftarrow\$ A.\mathrm{f}(z, param)$ ; Return $t$

Algorithm $\overline{\mathcal{M}}(1^k, pars)$
$(pk_n, sk_n, param) \leftarrow pars$
$(\mathbf{m}, \mathbf{a}) \leftarrow\$ \mathcal{M}(1^k, param)$
For $i = 1$ to $|\mathbf{m}|$ do
    $(a, xk, N) \leftarrow \mathbf{a}[i]$
    $\mathbf{y}[i] \leftarrow \mathsf{NE.Enc}(pk_n, xk, N, \mathbf{m}[i])$
Return $(\mathbf{y}, \mathbf{a})$

Algorithm $\overline{\mathsf{Rsmp}}(1^k, \mathbf{y}^*, \mathbf{a}^*, I, pars)$
$(pk_n, sk_n, param) \leftarrow pars$
For $i = 1$ to $|\mathbf{y}^*|$ do
    $\mathbf{m}^*[i] \leftarrow \mathsf{NE.Dec}(sk_n, \mathbf{y}^*[i])$
$z \leftarrow\$ \mathsf{Rsmp}(1^k, \mathbf{m}^*, \mathbf{a}^*, I, param)$
Return $z$
```

Figure 13: **D-SO-CPA2 adversary B, constructed sampler $\overline{\mathcal{M}}$, and its partial resampling algorithm $\overline{\mathsf{Rsmp}}$ in the proof of Theorem 5.1.**

secret keys for NE. It then runs $\mathcal{M}(1^k, param)$ to generate $(\mathbf{m}, \mathbf{a})$. Since $\mathcal{M}$ is unpredictable, each $\mathbf{a}[i]$ can be parsed as (a_i, xk_i, N_i). Now the "messages" of $\overline{\mathcal{M}}$ is the vector $\mathbf{y}$, where each $\mathbf{y}[i] = \mathsf{NE.Enc}(pk_n, xk_i, N_i, \mathbf{m}[i])$, and the corresponding auxiliary information is still $\mathbf{a}[i]$. The code of $\overline{\mathcal{M}}$ is given in Figure 13. Since $\mathcal{M}$ is (μ, d)-unpredictable, $\overline{\mathcal{M}}$ is $(\mathsf{NE.Guess}(\mu), d)$-entropic. Let δ be a function such that $\mathcal{M}$ is δ-partially resamplable. Let $\overline{\delta}(\mathbf{y}, pars)$ be the following function. It parses $pars$ as $(pk_n, sk_n, param)$, decrypts $\mathbf{m}[i] \leftarrow \mathsf{NE.Dec}(sk_n, \mathbf{y}[i])$, and then returns $\delta(\mathbf{m}, param)$. Then $\overline{\mathcal{M}}$ is $\overline{\delta}$-partially resamplable: given any δ-partial resampling algorithm Rsmp for $\mathcal{M}$, we can construct a $\overline{\delta}$-partial resampling algorithm $\overline{\mathsf{Rsmp}}$ for $\overline{\mathcal{M}}$ as in Figure 13.

Now, consider the adversary B attacking DE as given in Figure 13. It targets message sampler $\overline{\mathcal{M}}$, with respect to function $\overline{\delta}$. Initially, $B.\mathrm{pg}(1^k)$ runs $param \leftarrow A(1^k)$, and then generates public and secret keys pk_n and sk_n for NE. It then outputs $pars \leftarrow (pk_n, sk_n, param)$. When $B.\mathrm{g}$ receives its "messages" $\mathbf{y}^*$, it extracts the secret key sk_n from its state and decrypts $\mathbf{m}^*[i] \leftarrow \mathsf{NE.Dec}(sk_n, \mathbf{y}^*[i])$, and then gives $\mathbf{m}^*$ to $A.\mathrm{g}$ together with the auxiliary information $\mathbf{a}^*$. Then game $\mathrm{HN\text{-}CPA\text{-}REAL}^{A,\mathcal{M},\delta}_{\mathsf{NE}}$ coincides with game $\mathrm{D\text{-}CPA2\text{-}REAL}^{B,\overline{\mathcal{M}},\overline{\delta}}_{\mathsf{DE}}$. Moreover, game $\mathrm{HN\text{-}CPA\text{-}IDEAL}^{A,\mathcal{M},\delta}_{\mathsf{NE}}$ coincides with $\mathrm{D\text{-}CPA2\text{-}IDEAL}^{B,\overline{\mathcal{M}},\overline{\delta}}_{\mathsf{DE}}$. Hence $\mathbf{Adv}^{\text{hn-so-cpa}}_{\mathsf{NE},A,\mathcal{M},\delta}(\cdot) \le \mathbf{Adv}^{\text{d-so-cpa2}}_{\mathsf{DE},B,\overline{\mathcal{M}},\overline{\delta}}(\cdot)$. ∎

$\mathsf{NE1}$ ALONE IS ENOUGH. Constructions via NtD transform will be at least twice slower than $\mathsf{NE1}$, because we need to run public primitives twice. But in the NPROM, $\mathsf{NE1}[H, \mathsf{LT}]$ alone achieves both N-SO-CPA and HN-SO-CPA security simultaneously. In Theorem 5.2 below, we'll show that $\mathsf{NE1}$ is HN-SO-CPA secure. See Appendix B.4 for the proof. We stress that for (μ, ∞)-unpredictable message samplers, $\mathsf{NE1}$ allows the adversary to open as many ciphertexts as it wishes.

Theorem 5.2 Let LT be a lossy trapdoor function with lossiness τ. Let $\mathcal{M}$ be a (μ, d)-unpredictable resamplable message sampler, and let δ be a function such that $\mathcal{M}$ is δ-partially resamplable. Let $\mathsf{NE1}[H, \mathsf{LT}]$ be as above. In

```
$\underline{\mathsf{UE.Kg}(1^k)}$
$(pk, sk) \leftarrow_\$ \mathsf{DE.Kg}(1^k)$
Return $(pk, (pk, sk))$
```

```
$\underline{\mathsf{UE.Enc}(pk, m)}$
$c \leftarrow \mathsf{DE.Enc}(pk, m)$
Return $c$
```

```
$\underline{\mathsf{UE.Dec}((pk, sk), c)}$
$m \leftarrow \mathsf{DE.Dec}(sk, c)$
If $m \neq \perp$ then
    $c' \leftarrow \mathsf{DE.Enc}(pk, m)$
    If $c' \neq c$ then return $\perp$
Return $m$
```

Figure 14: **Unique-ciphertext D-PKE scheme** $\mathsf{UE} = \mathsf{UniqueCtx}[\mathsf{DE}]$ **constructed from** DE.

the NPROM, for any adversary A opening at most d ciphertexts, there is an adversary D such that

$$\mathbf{Adv}^{\text{hn-so-cpa}}_{\mathsf{NE1}[H,\mathsf{LT}],A,\mathcal{M},\delta}(k) \le \frac{4q(k)}{2^k} + \frac{4q(k)v(k)}{2^{\mu(k)}} + \frac{v(k)(v(k)+4q(k))}{2^{\tau(k)}} + 2\mathbf{Adv}^{\text{ltdf}}_{\mathsf{LT},D}(k),$$

where $q(k)$ is the total number of random-oracle queries of A and $\mathcal{M}$, and $v(k)$ is the number of messages that $\mathcal{M}$ produces. The running time of D is about that of A plus the time to run δ and an efficient δ-partial resampling algorithm of $\mathcal{M}$ plus the time to run $\mathsf{NE1}[H, \mathsf{LT}]$ to encrypt $\mathcal{M}$'s messages. Adversary D makes at most q random-oracle queries.

5.3 Achieving HN-SO-CCA Security

In proving that $\mathsf{NtD}[\mathsf{NE}, \mathsf{DE}]$ achieves HN-SO-CPA security, we don't need any property of the D-PKE scheme DE. This no longer holds for HN-SO-CCA. Indeed, consider a scheme $\mathsf{DE_{bad}}$ such that $\mathsf{DE_{bad}.Enc}$ appends 0 to the ciphertexts, and $\mathsf{DE_{bad}.Dec}$ ignores the last bit of the ciphertexts. An adversary thus can obtain the plaintexts by modifying the last bits of the ciphertexts, and querying those to the decryption oracle. Hence to obtain HN-SO-CCA, one has to exploit some property of DE. We'll need DE to be *unique-ciphertext*, a property formalized by Bellare and Hoang [7].

Formally, a D-PKE scheme DE is *unique-ciphertext* if for every $k \in \mathbb{N}$, every $(pk, sk) \in [\mathsf{DE.Kg}(1^k)]$, and every $m \in \{0,1\}^*$, there is at most a string c such that $\mathsf{DE.Dec}(sk, c) = m$. The D-PKE scheme $\mathsf{DE_{bad}}$ above is not unique-ciphertext. The unique-ciphertext property of DE ensures that if one modifies a ciphertext of $\mathsf{NtD}[\mathsf{NE}, \mathsf{DE}]$, the underneath ciphertext of NE will be changed.

Bellare and Hoang also show how to efficiently transform a D-PKE scheme DE to a unique-ciphertext one UE: in the decryption, we first recover the message, and then re-encrypt it and return $\perp$ if the newly constructed ciphertext doesn't match the given one. The transform $\mathsf{UniqueCtx}$ is given in Figure 14. Note that this transform doesn't affect the D-SO-CCA security of DE. Indeed, for any message sampler $\mathcal{M}$, any PT adversary A attacking $\mathsf{UE} = \mathsf{UniqueCtx}[\mathsf{DE}]$, it's trivial to construct another PT adversary B attacking DE such that $\mathbf{Adv}^{\text{d-so-cca}}_{\mathsf{UE},B,\mathcal{M}}(\cdot) \le \mathbf{Adv}^{\text{d-so-cca}}_{\mathsf{DE},A,\mathcal{M}}(\cdot)$.

Let DE be unique-ciphertext and D-SO-CCA secure D-PKE and NE be an N-SO-CCA secure, entropy-preserving nonce-based PKE. Then, Theorem 5.3 confirms $\mathsf{NtD}[\mathsf{NE}, \mathsf{DE}]$ achieves both HN-SO-CCA and N-SO-CCA security simultaneously; the proof is in Appendix B.5. To instantiate DE, one can either apply the $\mathsf{UniqueCtx}$ transform on a D-SO-CCA secure D-PKE scheme, or directly use our construction $\mathsf{DE2}$ in Section 3.3.

Theorem 5.3 Let NE be a nonce-based PKE as above, and let DE be a unique-ciphertext D-PKE scheme such that the ciphertext length of the former is a plaintext length of the latter. Let $\overline{\mathsf{NE}} = \mathsf{NtD}[\mathsf{NE}, \mathsf{DE}]$.

<u>N-SO-CCA security:</u> For any adversary A, any message sampler $\mathcal{M}$, any function δ such that $\mathcal{M}$ is δ-partially resamplable, and any nonce generator NG, there is an adversary B such that

$$\mathbf{Adv}^{\text{n-so-cca}}_{\overline{\mathsf{NE}},\mathsf{NG},A,\mathcal{M},\delta}(\cdot) \le \mathbf{Adv}^{\text{n-so-cca}}_{\mathsf{NE},\mathsf{NG},B,\mathcal{M},\delta}(\cdot) \ .$$

The running time of B is about that of A plus the running time of $\mathsf{DE.Kg}$ plus the time to run $\mathsf{DE.Enc}$ on the messages that A produces, and the time to run $\mathsf{DE.Dec}$ on the decryption queries of A. Adversary B makes as many decryption-oracle queries as A.

<u>HN-SO-CCA security:</u> For any adversary A, any (μ, d)-unpredictable message sampler $\mathcal{M}$, and any function δ such that $\mathcal{M}$ is δ-partially resamplable, there are an adversary B that opens the same number of ciphertexts, a function $\overline{\delta}$, and an $(\mathsf{NE.Guess}(\mu), d)$-entropic, $\overline{\delta}$-partially resamplable message sampler $\overline{\mathcal{M}}$ such that

$$\mathbf{Adv}^{\text{hn-so-cca}}_{\overline{\mathsf{NE}},A,\mathcal{M},\delta}(\cdot) \le \mathbf{Adv}^{\text{d-so-cca}}_{\mathsf{DE},B,\overline{\mathcal{M}},\overline{\delta}}(\cdot) \ .$$

The running time of B is about that of A plus the running time of $\overline{\mathsf{NE}}.\mathsf{Kg}$ plus the time to run $\overline{\mathsf{NE}}.\mathsf{Dec}$ on $v+p$ ciphertexts, where v is the number of messages that $\mathcal{M}$ produces and p is the number of A's decryption-oracle queries. Adversary B makes as many decryption-oracle queries as A. The running time of $\overline{\mathcal{M}}$ is about that of $\mathcal{M}$ plus the time to run $\mathsf{NE}.\mathsf{Enc}$ on v messages.

Alternatively, we can use $\mathsf{NE2}$ directly. In Theorem 5.4 below, we'll show that $\mathsf{NE2}$ is HN-SO-CCA secure. See Appendix B.6 for the proof.

Theorem 5.4 Let LT be a lossy trapdoor function with lossiness τ. Let $\mathcal{M}$ be a (μ, d)-unpredictable message sampler, and let δ be a function such that $\mathcal{M}$ is δ-partially resamplable. Let $\mathsf{NE2}[H, \mathsf{LT}]$ be as above. In the NPROM, for any adversary A opening at most d ciphertexts, there is an adversary D such that

$$\begin{aligned} & \mathbf{Adv}^{\text{hn-so-cca}}_{\mathsf{NE2}[H,\mathsf{LT}],A,\mathcal{M},\delta}(k) \\ \le \quad & \frac{6Q(k)}{2^k} + \frac{4Q(k)v(k)}{2^{\mu(k)}} + \frac{v(k)(v(k)+4Q(k))}{2^{\tau(k)}} + 2\mathbf{Adv}^{\text{ltdf}}_{\mathsf{LT},D}(k), \end{aligned}$$

where $q(k)$ is the total number of random-oracle queries of A and $\mathcal{M}$, $v(k)$ is the number of messages that $\mathcal{M}$ produces, and $p(k)$ is the number of decryption queries of A, and $Q(k) = q(k) + 2p(k)$. The running time of D is about that of A plus the time to run δ and a δ-partial resampling algorithm of $\mathcal{M}$ plus the time to run $\mathsf{NE2}[H, \mathsf{LT}]$ to encrypt $\mathcal{M}$'s messages. Adversary D makes at most Q random-oracle queries.

Acknowledgments

We thank the Asiacrypt reviewers for their insightful comments. This work was supported in part by NSF awards CNS-1223623, CNS-1423566, and CNS-1553758 (CAREER), as well as the Glen and Susanne Culler Chair. The work of Viet Tung Hoang was done while at the University of Maryland, Georgetown University, and UC Santa Barbara.

References

[1] B. Barak, Y. Dodis, H. Krawczyk, O. Pereira, K. Pietrzak, F.-X. Standaert, and Y. Yu. Leftover hash lemma, revisited. In P. Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 1–20, Santa Barbara, CA, USA, Aug. 14–18, 2011. Springer, Berlin, Germany. (Cited on page 17.)

[2] M. Bellare, A. Boldyreva, and A. O'Neill. Deterministic and efficiently searchable encryption. In A. Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 535–552, Santa Barbara, CA, USA, Aug. 19–23, 2007. Springer, Berlin, Germany. (Cited on page 3, 4, 6.)

[3] M. Bellare, Z. Brakerski, M. Naor, T. Ristenpart, G. Segev, H. Shacham, and S. Yilek. Hedged public-key encryption: How to protect against bad randomness. In M. Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 232–249, Tokyo, Japan, Dec. 6–10, 2009. Springer, Berlin, Germany. (Cited on page 3, 4.)

[4] M. Bellare, R. Dowsley, and S. Keelveedhi. How secure is deterministic encryption? In *Public-Key Cryptography — PKC 2015*, volume 9020 of *LNCS*, pages 52–73. Springer, Berlin, Germany, 2015. (Cited on page 4, 6, 8, 13.)

[5] M. Bellare, R. Dowsley, B. Waters, and S. Yilek. Standard security does not imply security against selective-opening. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 645–662, Cambridge, UK, Apr. 15–19, 2012. Springer, Berlin, Germany. (Cited on page 3.)

[6] M. Bellare, M. Fischlin, A. O'Neill, and T. Ristenpart. Deterministic encryption: Definitional equivalences and constructions without random oracles. In D. Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 360–378, Santa Barbara, CA, USA, Aug. 17–21, 2008. Springer, Berlin, Germany. (Cited on page 4, 6.)

[7] M. Bellare and V. T. Hoang. Resisting randomness subversion: Fast deterministic and hedged public-key encryption in the standard model. In *Advances in Cryptology—EUROCRYPT 2015*, volume 9057 of *LNCS*, pages 627–656. Springer, Berlin, Germany, 2015. (Cited on page 3, 4, 8, 9, 10, 19.)

[8] M. Bellare, V. T. Hoang, and S. Keelveedhi. Instantiating random oracles via UCEs. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 398–415, Santa Barbara, CA, USA, Aug. 18–22, 2013. Springer, Berlin, Germany. (Cited on page 4, 21, 22.)

[9] M. Bellare, D. Hofheinz, and S. Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In A. Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 1–35, Cologne, Germany, Apr. 26–30, 2009. Springer, Berlin, Germany. (Cited on page 3, 4, 5, 6, 13.)

[10] M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Berlin, Germany. (Cited on page 26.)

[11] M. Bellare and B. Tackmann. Nonce-based cryptography: Retaining security when randomness fails. In *Advances in Cryptology—EUROCRYPT 2016*, 2016. (Cited on page 3, 4, 11, 12, 13.)

[12] F. Böhl, D. Hofheinz, and D. Kraschewski. On definitions of selective opening security. In M. Fischlin, J. Buchmann, and M. Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 522–539, Darmstadt, Germany, May 21–23, 2012. Springer, Berlin, Germany. (Cited on page 3.)

[13] A. Boldyreva, S. Fehr, and A. O'Neill. On notions of security for deterministic encryption, and efficient constructions without random oracles. In D. Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 335–359, Santa Barbara, CA, USA, Aug. 17–21, 2008. Springer, Berlin, Germany. (Cited on page 21.)

[14] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart. Leakage-abuse attacks against searchable encryption. In *CCS*, pages 668–679, 2015. (Cited on page 11.)

[15] E. Fujisaki and T. Okamoto. How to enhance the security of public-key encryption at minimum cost. In H. Imai and Y. Zheng, editors, *PKC'99*, volume 1560 of *LNCS*, pages 53–68, Kamakura, Japan, Mar. 1–3, 1999. Springer, Berlin, Germany. (Cited on page 9.)

[16] F. Heuer, E. Kiltz, and K. Pietrzak. Standard security does imply security against selective opening for markov distributionss. Cryptology ePrint Archive, Report 2015/853, 2015. `http://eprint.iacr.org/2015/853`. (Cited on page 3.)

[17] D. Hofheinz, V. Rao, and D. Wichs. Standard security does not imply indistinguishability under selective opening. Cryptology ePrint Archive, Report 2015/792, 2015. `http://eprint.iacr.org/2015/792`. (Cited on page 3, 4, 6, 13, 21.)

[18] D. Hofheinz and A. Rupp. Standard versus selective opening security: Separation and equivalence results. In Y. Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 591–615, San Diego, CA, USA, Feb. 24–26, 2014. Springer, Berlin, Germany. (Cited on page 3.)

[19] Y. Ishai, O. Pandey, and A. Sahai. Public-coin differing-inputs obfuscation and its applications. In *Theory of Cryptography Conference*, pages 668–697. Springer, 2015. (Cited on page 13.)

[20] S. Kamara and J. Katz. How to encrypt with a malicious random number generator. In K. Nyberg, editor, *FSE 2008*, volume 5086 of *LNCS*, pages 303–315, Lausanne, Switzerland, Feb. 10–13, 2008. Springer, Berlin, Germany. (Cited on page 3.)

[21] E. Kiltz, A. O'Neill, and A. Smith. Instantiability of RSA-OAEP under chosen-plaintext attack. In T. Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 295–313, Santa Barbara, CA, USA, Aug. 15–19, 2010. Springer, Berlin, Germany. (Cited on page 5, 8.)

[22] J. B. Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In M. Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 111–126, Santa Barbara, CA, USA, Aug. 18–22, 2002. Springer, Berlin, Germany. (Cited on page 3.)

[23] C. Peikert and B. Waters. Lossy trapdoor functions and their applications. In R. E. Ladner and C. Dwork, editors, *40th ACM STOC*, pages 187–196, Victoria, British Columbia, Canada, May 17–20, 2008. ACM Press. (Cited on page 3, 5, 8.)

[24] Y. Seurin. On the lossiness of the Rabin trapdoor function. In H. Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 380–398, Buenos Aires, Argentina, Mar. 26–28, 2014. Springer, Berlin, Germany. (Cited on page 5, 8.)

[25] S. Yilek. Resettable public-key encryption: How to encrypt on a virtual machine. In J. Pieprzyk, editor, *CT-RSA 2010*, volume 5985 of *LNCS*, pages 41–56, San Francisco, CA, USA, Mar. 1–5, 2010. Springer, Berlin, Germany. (Cited on page 3.)

A Standard Security Doesn't Imply SOA on D-PKE

In this appendix, we'll show that standard security notions for D-PKE don't imply D-SO-CPA1. We'll use the IND notion of Boldyreva, Fehr, and O'Neill [13], which is equivalent to PRIV but more convenient to work with. Below, we'll recall the IND notion. We also recall the UCE notion of hash functions [8], which will be used to transform the recent R-PKE scheme of Hofheinz, Rao, and Wichs (HRW) [17] to a D-PKE one.

```
Game IND-CPA$^B_{\mathsf{RE}}(k)$
    $(pk, sk) \leftarrow\$ \mathsf{RE.Kg}(1^k)$ ; $b \leftarrow\$ \{0,1\}$
    $b' \leftarrow\$ A^{\mathrm{LR}}(1^k, pk)$ ; Return $(b = b')$

Procedure LR$(m_0, m_1)$
    Return $\mathsf{RE.Enc}(pk, m_b)$
```

Figure 15: **Game defining IND-CPA security.** For queries (m_0, m_1) to the oracle LR, we require that $|m_0| = |m_1|$

```
Game IND$^A_{\mathsf{DE}}(k)$
$b \leftarrow\$ \{0,1\}$
$(pk, sk) \leftarrow\$ \mathsf{DE.Kg}(1^k)$
$(\mathbf{m}_0, \mathbf{m}_1) \leftarrow\$ A_1(1^k)$
For $i = 1$ to $|\mathbf{m}_0|$ do
    $\mathbf{c}[i] \leftarrow \mathsf{DE.Enc}(pk, \mathbf{m}_b[i])$
$b' \leftarrow\$ A_2(1^k, pk, \mathbf{c})$
Return $(b = b')$

Game UCE$^{S,D}_{\mathsf{H}}(k)$
$b \leftarrow\$ \{0,1\}$ ; $hk \leftarrow\$ \mathsf{HKg}(1^k)$
$L \leftarrow\$ S^{\mathrm{HASH}}(1^k)$ ; $b' \leftarrow\$ D(1^k, hk, L)$
Return $(b' = b)$

Procedure HASH$(x, \ell)$
If $T[x, \ell] = \bot$ then
    If $b = 0$ then $T[x, \ell] \leftarrow\$ \{0,1\}^\ell$
    Else $T[x, \ell] \leftarrow \mathsf{h}(hk, x, \ell)$
Return $T[x, \ell]$

Game Pred$^P_S(k)$
$Q \leftarrow \emptyset$ ; $L \leftarrow\$ S^{\mathrm{HASH}}(1^k)$
$x \leftarrow\$ P(1^k, L)$
Return $x \in Q$

Procedure HASH$(x, \ell)$
If $T[x, \ell] = \bot$ then $T[x, \ell] \leftarrow\$ \{0,1\}^\ell$
$Q \leftarrow Q \cup \{x\}$
Return $T[x, \ell]$
```

Figure 16: **Game** IND **(left) defining security of a D-PKE** DE**, and games** UCE **(middle) and** Pred **(right) defining UCE security of a hash function family** H**.**

A.1 Some Security Notions

IND-CPA. Let $\mathsf{RE} = (\mathsf{RE.Kg}, \mathsf{RE.Enc}, \mathsf{RE.Dec})$ be a randomized PKE. We say that RE is IND-CPA-secure if for any PT adversary A,

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{RE},A}(\cdot) = 2\Pr[\text{IND-CPA}(\cdot)] - 1$$

is negligible, where game IND-CPA is defined in Figure 15.

IND SECURITY. The formal definition of IND for a D-PKE scheme $\mathsf{DE} = (\mathsf{DE.Kg}, \mathsf{DE.Enc}, \mathsf{DE.Dec})$ is shown in the left panel of Figure 16. An IND adversary A is a pair (A_1, A_2). Adversary $A_1(1^k)$ first outputs a pair of message vectors $(\mathbf{m}_0, \mathbf{m}_1)$. We require that (i) there is a polynomial v such that $|\mathbf{m}_0| = |\mathbf{m}_1| = v(k)$, (ii) there is a polynomial n such that $|\mathbf{m}_b[i]| = n(k)$, for every $b \in \{0,1\}$ and every $i \in \{1, \ldots, v(k)\}$, and (iii) for each $b \in \{0,1\}$, the messages $\mathbf{m}_b[1], \ldots, \mathbf{m}_b[v(k)]$ are distinct. Let $\mathrm{Guess}_A(k) = \max_{b,m,i}\{\Pr[\mathbf{m}_b[i] = m : (\mathbf{m}_0, \mathbf{m}_1) \leftarrow\$ A_1(1^k)]\}$. An adversary A has *high min-entropy* if Guess_A is a negligible function. The game then picks $b \leftarrow\$ \{0,1\}$, generates the public key $pk \leftarrow\$ \mathsf{DE.Kg}(1^k)$, and encrypts $\mathbf{c}[i] \leftarrow \mathsf{DE.Enc}(pk, \mathbf{m}[i])$ for every $i \in \{1, \ldots, v(k)\}$. Adversary A_2 is given pk and $\mathbf{c}$, and has to guess the challenge bit b. Define $\mathbf{Adv}^{\text{ind}}_{\mathsf{DE},A}(\cdot) = 2\Pr[\mathrm{IND}^A_{\mathsf{DE}}(\cdot) \Rightarrow 1] - 1$. The scheme DE is IND-secure if $\mathbf{Adv}^{\text{ind}}_{\mathsf{DE},A}(\cdot)$ is negligible for every PT adversary A that has high min-entropy.

UCE. We now recall the UCE framework of Bellare, Hoang, and Keelveedhi [8]. Let $\mathsf{H} = (\mathsf{HKg}, \mathsf{h})$ be a family of hash functions of variable input and output length. That is, the function h takes as input a key hk, a string x, and an integer ℓ, and then outputs a string $y = \mathsf{h}(hk, x, \ell) \in \{0,1\}^\ell$. Let S be a source, and D be a distinguisher. The game $\mathrm{UCE}^{S,D}_{\mathsf{H}}$ in which S and D jointly attack H is shown in Figure 16. The game starts by choosing a bit $b \leftarrow\$ \{0,1\}$ and a hash key $hk \leftarrow\$ \mathsf{HKg}(1^k)$. The source is given access to an oracle HASH that implements $\mathsf{h}(hk, \cdot, \cdot)$ if $b = 1$, and implements a random oracle (of arbitrary input and output length) otherwise. The source then leaks some information L to the distinguisher. The latter is also given the hash key hk, and has to guess the challenge bit b. Define $\mathbf{Adv}^{\text{uce}}_{\mathsf{H},S,D}(\cdot) = 2\Pr[\mathrm{UCE}^{S,D}_{\mathsf{H}}(\cdot)] - 1$.

To avoid trivial attacks, we have to restrict the source. There are many ways to do it, but here we'll focus on *statistically unpredictable* sources. The unpredictability of a source S is measured via game Pred in Figure 16. Note that this game doesn't involve the hash family H. In this game, the source is again given access to an oracle HASH, but this time the oracle always implements a random oracle. The leakage L is now given to a predictor P who tries to guess one of the source's queries. Define $\mathbf{Adv}^{\text{pred}}_{S,P}(\cdot) = \Pr[\mathrm{Pred}^P_S(\cdot)]$. A source is statistically unpredictable if $\mathbf{Adv}^{\text{pred}}_{S,P}(\cdot)$ is negligible for all (including computationally unbounded) predictors.

The hash family H is UCE-secure if $\mathbf{Adv}^{\text{uce}}_{\mathsf{H},S,D}(\cdot)$ is negligible for all PT distinguishers D and all PT statistically unpredictable sources S.

A.2 Separating IND and D-SO-CPA1

HRW CONSTRUCTION. Our counterexample is based on the recent (contrived) construction $\mathsf{RE}_{\mathsf{bad}} = (\mathsf{RE}_{\mathsf{bad}}.\mathsf{Kg},$

```
$\underline{\mathsf{DE_{bad}.Kg}(1^k)}$
$hk \leftarrow\$ \mathsf{HKg}(1^k)$
$(ek, td) \leftarrow\$ \mathsf{LT.IKg}(1^k)$
$(pk', sk') \leftarrow\$ \mathsf{RE_{bad}.Kg}(1^k)$
$pk \leftarrow (hk, ek, pk')$
$sk \leftarrow (hk, td, sk'))$
Return $(pk, sk)$
```

```
$\underline{\mathsf{DE_{bad}.Enc}(pk, m)}$
$(hk, ek, pk') \leftarrow pk$ ; $x \leftarrow m[1, \ell(k)]$
$r \leftarrow \mathsf{h}(hk, 00 \,\|\, m, \mathsf{LT.il}(k))$
$R \leftarrow \mathsf{h}(hk, 01 \,\|\, m, \mathsf{RE_{bad}.rl}(k))$
$trap \leftarrow \mathsf{LT.Eval}(ek, r)$
$y \leftarrow \mathsf{h}(hk, 10 \,\|\, r, |m|) \oplus m$
$z \leftarrow \mathsf{RE_{bad}.Enc}(pk', x; R)$
Return $(trap, y, z)$
```

```
$\underline{\mathsf{DE_{bad}.Dec}(sk, c)}$
$(trap, y, z) \leftarrow c$ ; $(hk, td, sk') \leftarrow sk$
$r \leftarrow \mathsf{LT.Inv}(td, trap)$
$m \leftarrow \mathsf{h}(hk, 1 \,\|\, r, |y|) \oplus y$
Return $m$
```

Figure 17: **D-PKE $\mathsf{DE_{bad}}$ that is IND-secure but is not D-SO-CPA1-secure.**

$\mathsf{RE_{bad}.Enc}, \mathsf{RE_{bad}.Dec}$) of HRW to separate IND-CPA and SOA security of R-PKE. The scheme $\mathsf{RE_{bad}}$ is IND-CPA secure, but is vulnerable to the following SOA attack. The message sampler $\mathcal{M}(1^k, param)$ ignores $param$, picks a secret $s \leftarrow\$ \{0,1\}^\ell$ and then secret-shares it to $v(k)$ messages $\mathbf{m}[1], \ldots, \mathbf{m}[v(k)]$ so that any $t(k)$ shares reveal no information of the secret s. In other words, it picks $a_0, a_1, \ldots, a_t$ uniformly from $\mathsf{GF}(2^\ell)$, the finite field of size 2^ℓ, and computes $\mathbf{m}[i] \leftarrow f(i)$ for every $i \in \{1, \ldots, v(k)\}$, where $f(x) = a_0 + a_1 x + \cdots + a_t x^t$ is the corresponding polynomial in $\mathsf{GF}(2^\ell)[X]$. Recall that any $t+1$ shares will uniquely determine the polynomial f (via polynomial interpolation), and thus any $t+1$ shares are uniformly and independently random. Still, there's an efficient adversary that opens only t ciphertexts and can recover all messages.

Our D-PKE scheme. Our D-PKE scheme $\mathsf{DE_{bad}}[\mathsf{LT}, \mathsf{H}, \mathsf{RE_{bad}}] = (\mathsf{DE_{bad}.Kg}, \mathsf{DE_{bad}.Enc}, \mathsf{DE_{bad}.Dec})$, shown in Figure 17, is based on the R-PKE scheme $\mathsf{RE_{bad}}$ above, a UCE-secure hash family H, and a lossy trapdoor function LT. It accepts messages of length $3\ell(k)$, and requires that each individual message must have entropy at least $2\ell(k)$. On a message m, it will essentially use the D-SO-CPA1-secure scheme in Section 3.2 to encrypt m, and uses $\mathsf{RE_{bad}}$ to encrypt the first ℓ bits of m; the coins of $\mathsf{RE_{bad}}$ is produced via hashing m.

Results. Following HRW's result, we can launch a D-SO-CPA1 attack to $\mathsf{DE_{bad}}$ as follows. The attack doesn't need a common parameter $param$; we therefore omit A.pg and also the argument $param$ and the auxiliary information. The message sampler $\mathcal{M}(1^k)$ first picks a secret $s \leftarrow\$ \{0,1\}^{\ell(k)}$ and secret-shares it to $\mathbf{x}[1], \ldots, \mathbf{x}[v(k)]$ such that any t shares reveal no information about s. It then chooses $\mathbf{u}[1], \ldots, \mathbf{u}[v(k)] \leftarrow\$ \{0,1\}^{2\ell(k)}$, and outputs $\mathbf{m}[1], \ldots, \mathbf{m}[v(k)]$, where each $\mathbf{m}[i]$ is $\mathbf{x}[i] \,\|\, \mathbf{u}[i]$. This message sampler is $(3\ell, t)$-entropic and also efficiently resamplable. Indeed, given $\mathbf{m}[I]$, we can resample the vector $\mathbf{m}$ as follows.

- **Case 1:** $|I| \geq t+1$. Let $\mathbf{m}'[I] = \mathbf{m}[I]$, and let $\mathbf{x}[i]$ be the first ℓ bits of $\mathbf{m}'[i]$, for every $i \in I$. Then $\mathbf{x}[I]$ uniquely determine a polynomial f in $\mathsf{GF}(2^\ell)[X]$ such that $\mathbf{x}[i] = f(i)$ for every $i \in I$. For every $j \in \{1, \ldots, v(k)\} \backslash I$, let $\mathbf{m}'[j] \leftarrow \mathbf{x}[j] \,\|\, \mathbf{u}'[j]$, where $\mathbf{u}'[j] \leftarrow\$ \{0,1\}^{2\ell}$ and $\mathbf{x}[j] \leftarrow f(j)$.
- **Case 2:** $|I| \leq t$. Again let $\mathbf{m}'[I] \leftarrow \mathbf{m}[I]$. Let S be a subset of $\{1, \ldots, v(k)\} \backslash I$ such that $|S| = t + 1 - |I|$. Sample $\mathbf{m}'[i] \leftarrow\$ \{0,1\}^{3\ell}$ for every $i \in S$. Following Case 1 above to resample $\mathbf{m}'$ given $\mathbf{m}'[I \cup S]$.

Now, using HRW's result, we can build PT adversaries (A.cor, A.g) that open t ciphertexts and recover $\mathbf{x}$, where each $\mathbf{x}[i]$ is the first ℓ bits of $\mathbf{m}[i]$. Adversary A.g then outputs the checksum $\mathbf{x}[1] \oplus \cdots \oplus \mathbf{x}[v(k)]$. Likewise, adversary A.f, on input $\mathbf{m}^*$, parses each $\mathbf{m}^*[i]$ as $\mathbf{x}^*[i] \,\|\, \mathbf{u}^*[i]$, where $|\mathbf{x}^*[i]| = \ell$, and outputs $\mathbf{x}^*[1] \oplus \cdots \oplus \mathbf{x}^*[v(k)]$. The adversary A thus wins with advantage $1 - 2^{-\ell(k)}$.

What's left is to show that $\mathsf{DE_{bad}}$ is indeed IND-secure, which is confirmed by Proposition A.1 below.

Proposition A.1 Let H be a UCE-secure hash family and LT be a lossy trapdoor function with lossiness τ. Let $\mathsf{RE_{bad}}$ and $\mathsf{DE_{bad}}[\mathsf{LT}, \mathsf{H}, \mathsf{RE_{bad}}]$ be as above. Then for any adversary A such that $\mathrm{Guess}_A(\cdot) \leq 2^{-2\ell}$, there are adversaries B_1, B_2, source S, and distinguisher D such that $\mathbf{Adv}^{\mathrm{ind}}_{\mathsf{DE_{bad}}[\mathsf{LT},\mathsf{H},\mathsf{RE_{bad}}]}(\cdot) \leq 2\mathbf{Adv}^{\mathrm{uce}}_{\mathsf{H},S,D}(\cdot) + 2\mathbf{Adv}^{\mathrm{ltdf}}_{\mathsf{LT},B_1}(\cdot) + \mathbf{Adv}^{\mathrm{cpa}}_{\mathsf{RE_{bad}},B_2}(\cdot) + v^2/\mathsf{LT.il}$, where v is the number of components in a message vector of A. Moreover, for any predictor P, it holds that $\mathbf{Adv}^{\mathrm{pred}}_{S,P}(\cdot) \leq v/2^\tau + v/2^\ell + v^2/2^{\mathsf{LT.il}}$. The running time of each of B_1, B_2, S, D is about that of A plus the time to use $\mathsf{DE_{bad}}$ to encrypt A's messages.

Proof: Consider games G_1–G_5 in Figure 18. Game G_1 corresponds to game $\mathrm{IND}^A_{\mathsf{DE_{bad}}[\mathsf{LT},\mathsf{H},\mathsf{RE_{bad}}]}$. Then

$$\mathbf{Adv}^{\mathrm{ind}}_{\mathsf{DE_{bad}}[\mathsf{LT},\mathsf{H},\mathsf{RE_{bad}}],A}(\cdot) = 2\Pr[G_1(\cdot)] - 1 \ .$$

```
**Games** $G_1(k)$, $\boxed{G_2(k)}$
$(\mathbf{m}_0, \mathbf{m}_1) \leftarrow\$ A_1(1^k)$ ; $b \leftarrow\$ \{0,1\}$
$(ek, td) \leftarrow\$ \mathsf{LT.IKg}(1^k)$
$\boxed{ek \leftarrow\$ \mathsf{LT.LKg}(1^k)\ ;}$  $hk \leftarrow \mathsf{HKg}(1^k)$
$(pk', sk') \leftarrow\$ \mathsf{RE_{bad}.Kg}(1^k)$
For $i = 1$ to $v(k)$ do
    $m \leftarrow \mathbf{m}_b[i]$ ; $\mathbf{x}[i] \leftarrow m[1, \ell(k)]$
    $\mathbf{r}[i] \leftarrow \mathsf{h}(hk, 00 \,\|\, m, \mathsf{LT.il}(k))$
    $R \leftarrow \mathsf{h}(hk, 01 \,\|\, m, \mathsf{RE_{bad}.rl}(k))$
    $trap \leftarrow \mathsf{LT.Eval}(ek, \mathbf{r}[i])$
    $y \leftarrow \mathsf{h}(hk, 10 \,\|\, \mathbf{r}[i], |m|) \oplus m$
    $z \leftarrow \mathsf{RE_{bad}.Enc}(pk', \mathbf{x}[i]; R)$
    $\mathbf{c}[i] \leftarrow (trap, y, z)$
$b' \leftarrow\$ A_2(1^k, (hk, ek, pk'), \mathbf{c})$
Return $(b = b')$
```

```
**Games** $\boxed{G_3(k),}$ $G_4(k)$
$(\mathbf{m}_0, \mathbf{m}_1) \leftarrow\$ A_1(1^k)$ ; $b \leftarrow\$ \{0,1\}$
$ek \leftarrow\$ \mathsf{LT.LKg}(1^k)$ ; $hk \leftarrow \mathsf{HKg}(1^k)$
$(pk', sk') \leftarrow\$ \mathsf{RE_{bad}.Kg}(1^k)$
For $i = 1$ to $v(k)$ do
    $m \leftarrow \mathbf{m}_b[i]$ ; $\mathbf{x}[i] \leftarrow m[1, \ell(k)]$
    $\mathbf{r}[i] \leftarrow\$ \{0,1\}^{\mathsf{LT.il}(k)}$
    $R \leftarrow\$ \{0,1\}^{\mathsf{RE_{bad}.rl}(k)}$
    $trap \leftarrow \mathsf{LT.Eval}(ek, \mathbf{r}[i])$
    $pad \leftarrow\$ \{0,1\}^{|m|}$
    If $H[\mathbf{r}[i]] \neq \bot$ then
        $\mathsf{bad} \leftarrow \mathsf{true}$ ; $\boxed{pad \leftarrow H[\mathbf{r}[i]]}$
    $H[\mathbf{r}[i]] \leftarrow pad$
    $z \leftarrow \mathsf{RE_{bad}.Enc}(pk', \mathbf{x}[i]; R)$
    $y \leftarrow pad \oplus m$ ; $\mathbf{c}[i] \leftarrow (trap, y, z)$
$b' \leftarrow\$ A_2(1^k, (hk, ek, pk'), \mathbf{c})$
Return $(b = b')$
```

```
**Game** $G_5(k)$
$(\mathbf{m}_0, \mathbf{m}_1) \leftarrow\$ A_1(1^k)$ ; $b \leftarrow\$ \{0,1\}$
$ek \leftarrow\$ \mathsf{LT.LKg}(1^k)$ ; $hk \leftarrow \mathsf{HKg}(1^k)$
$(pk', sk') \leftarrow\$ \mathsf{RE_{bad}.Kg}(1^k)$
For $i = 1$ to $v(k)$ do
    $m \leftarrow \mathbf{m}_b[i]$ ; $\mathbf{x}[i] \leftarrow m[1, \ell(k)]$
    $\mathbf{r}[i] \leftarrow\$ \{0,1\}^{\mathsf{LT.il}(k)}$
    $R \leftarrow\$ \{0,1\}^{\mathsf{RE_{bad}.rl}(k)}$
    $trap \leftarrow \mathsf{LT.Eval}(ek, \mathbf{r}[i])$
    $z \leftarrow \mathsf{RE_{bad}.Enc}(pk', \mathbf{x}[i]; R)$
    $y \leftarrow\$ \{0,1\}^{|m|}$ ; $\mathbf{c}[i] \leftarrow (trap, y, z)$
$b' \leftarrow\$ A_2(1^k, (hk, ek, pk'), \mathbf{c})$
Return $(b = b')$
```

Figure 18: **Games G_1–G_5 in the proof of Proposition A.1.** Games G_2 and G_3 contain the corresponding boxed statements, but games G_1 and G_4 do not.

```
**Algorithm** $S^{\text{HASH}}(1^k)$
$(\mathbf{m}_0, \mathbf{m}_1) \leftarrow\$ A_1(1^k)$ ; $b \leftarrow\$ \{0,1\}$
$ek \leftarrow\$ \mathsf{LT.LKg}(1^k)$
$(pk', sk') \leftarrow\$ \mathsf{RE_{bad}.Kg}(1^k)$
For $i = 1$ to $v(k)$ do
    $m \leftarrow \mathbf{m}_b[i]$ ; $\mathbf{x}[i] \leftarrow m[1, \ell(k)]$
    $\mathbf{r}[i] \leftarrow \text{HASH}(00 \,\|\, m, \mathsf{LT.il}(k))$
    $R \leftarrow \text{HASH}(01 \,\|\, m, \mathsf{RE_{bad}.rl}(k))$
    $trap \leftarrow \mathsf{LT.Eval}(ek, \mathbf{r}[i])$
    $y \leftarrow \text{HASH}(10 \,\|\, \mathbf{r}[i], |m|) \oplus m$
    $z \leftarrow \mathsf{RE_{bad}.Enc}(pk', \mathbf{x}[i]; R)$
    $\mathbf{c}[i] \leftarrow (trap, y, z)$
$L \leftarrow (b, ek, pk', \mathbf{c})$ ;  Return $L$
```

```
**Algorithm** $D(1^k, hk, L)$
$(b, ek, pk', \mathbf{c}) \leftarrow L$
$b' \leftarrow\$ A_2(1^k, (hk, ek, pk'), \mathbf{c})$
If $(b' = b)$ then return 1
Else return 0
```

```
**Algorithm** $B_2^{\text{LR}}(1^k, pk')$
$(\mathbf{m}_0, \mathbf{m}_1) \leftarrow\$ A_1(1^k)$ ; $b \leftarrow\$ \{0,1\}$
$ek \leftarrow\$ \mathsf{LT.LKg}(1^k)$ ; $hk \leftarrow \mathsf{HKg}(1^k)$
For $i = 1$ to $v(k)$ do
    $m \leftarrow \mathbf{m}_b[i]$ ; $\mathbf{x}[i] \leftarrow m[1, \ell(k)]$
    $\mathbf{r}[i] \leftarrow\$ \{0,1\}^{\mathsf{LT.il}(k)}$
    $trap \leftarrow \mathsf{LT.Eval}(ek, \mathbf{r}[i])$
    $z \leftarrow \text{LR}(\mathbf{x}[i])$
    $y \leftarrow\$ \{0,1\}^{|m|}$ ; $\mathbf{c}[i] \leftarrow (trap, y, z)$
$b' \leftarrow\$ A_2(1^k, (hk, ek, pk'), \mathbf{c})$
If $(b = b')$ then return 1
Else return 0
```

Figure 19: **The source S (left), distinguisher D (middle), and adversary B_2 (right) in the proof of Proposition A.1.**

We now explain the game chain. Game G_2 is identical to game G_1, except that instead of using an injective key for the lossy trapdoor function, we use a lossy key. Consider the following adversary B_1 attacking the key indistinguishability of LT. It simulates game G_1, but uses its given key instead of generating a new one. Then

$$\Pr[G_1(\cdot) \Rightarrow 1] - \Pr[G_2(\cdot) \Rightarrow] = \mathbf{Adv}^{\text{ltdf}}_{\mathsf{LT},B_1}(\cdot) \ .$$

Next, game G_3 is identical to game G_2, except that instead of using h to hash, we'll use a random oracle. Consider the source S and distinguisher D in Figure 19. They collaborate to simulate game G_2, but the source uses the oracle HASH instead of using the hash h. The distinguisher returns 1 if adversary A_2 can correctly guess the simulated challenge bit b, and returns 0 otherwise. Then

$$\Pr[G_2(\cdot) \Rightarrow 1] - \Pr[G_3(\cdot) \Rightarrow 1] = \mathbf{Adv}^{\text{uce}}_{\mathsf{H},S,D}(\cdot) \ .$$

In game G_4, when we apply the random oracle on the synthetic coins $\mathbf{r}$, if some $\mathbf{r}[i]$ collides with a prior $\mathbf{r}[j]$ then we ignore the consistency, producing a fresh random output. From the PRP/PRF Switching Lemma,

$$\Pr[G_3(\cdot) \Rightarrow 1] - \Pr[G_4(\cdot) \Rightarrow 1] \leq \Pr[G_4(\cdot) \text{ sets } \mathsf{bad}] \leq \frac{v^2}{2^{1+\mathsf{LT.il}}} \ .$$

```
Game $G_1(k)$
$hk \leftarrow\$ \{0,1\}^k$ ; $param \leftarrow\$ A.\mathrm{pg}^{\mathrm{RO}_1}(1^k)$
$(\mathbf{m}, \mathbf{a}) \leftarrow\$ \mathcal{M}^{\mathrm{RO}_1}(1^k, param)$ ; $z_1 \leftarrow \delta(\mathbf{m}, param)$ ; $(ek, td) \leftarrow\$ \mathsf{LT.IKg}(1^k)$
For $i = 1$ to $|\mathbf{m}|$ do
    $\mathbf{r}[i] \leftarrow \mathrm{RO}(hk \,\|\, 0 \,\|\, \mathbf{m}[i], \mathsf{LT.il}(k))$ ; $trap \leftarrow \mathsf{LT.Eval}(ek, \mathbf{r}[i])$
    $y \leftarrow \mathrm{RO}(hk \,\|\, 1 \,\|\, \mathbf{r}[i], |\mathbf{m}[i]|) \oplus \mathbf{m}[i]$ ; $\mathbf{c}[i] \leftarrow (trap, y)$
$(state, I) \leftarrow\$ A.\mathrm{cor}^{\mathrm{RO}_2}((hk, ek), \mathbf{c}, param)$ ; $\omega \leftarrow\$ A.\mathrm{g}^{\mathrm{RO}_3}(state, \mathbf{m}[I], \mathbf{a}[I])$
$z_0 \leftarrow\$ \mathsf{Rsmp}^{\mathrm{RO}_1}(1^k, \mathbf{m}[I], \mathbf{a}[I], I, param)$ ; $b \leftarrow\$ \{0,1\}$ ; $t \leftarrow\$ A.\mathrm{f}^{\mathrm{RO}_4}(z_b, param)$
If $(\omega = t)$ then return $b$ else return $1 - b$

Procedure $\mathrm{RO}_1(x, \ell)$
Return $\mathrm{RO}(x, \ell)$

Procedure $\mathrm{RO}_2(x, \ell)$
Return $\mathrm{RO}(x, \ell)$

Procedure $\mathrm{RO}_3(x, \ell)$
Return $\mathrm{RO}(x, \ell)$

Procedure $\mathrm{RO}_4(x, \ell)$
Return $\mathrm{RO}(x, \ell)$
```

Figure 20: **Game G_1 of the proof of Theorem 3.2.**

Game G_5 is a simplification of game G_4. Since we always use a fresh, truly random one-time pad to mask each message $\mathbf{m}_b[i]$, we now instead pick the masked string uniformly at random. Then

$$\Pr[G_4(\cdot) \Rightarrow 1] = \Pr[G_5(\cdot) \Rightarrow 1] \ .$$

Next, consider the adversary B_2 attacking $\mathsf{RE}_{\mathsf{bad}}$ in Figure 19. It simulates game G_5, but uses its given public key instead of generating a new key pk' for $\mathsf{RE}_{\mathsf{bad}}$. Moreover, it uses its oracle to encrypt the first ℓ bits of each message $\mathbf{m}_b[i]$. It returns 1 if adversary A_2 can correctly guess the simulated challenge bit b, and returns 0 otherwise. Then

$$\Pr[G_5(\cdot) \Rightarrow 1] = \Pr[\mathrm{CPA}^{B_2}_{\mathsf{RE}_{\mathsf{bad}}}(\cdot) \Rightarrow 1] \ .$$

Summing up,

$$\mathbf{Adv}^{\mathrm{ind}}_{\mathsf{DE}_{\mathsf{bad}}[\mathsf{LT},\mathsf{H},\mathsf{RE}_{\mathsf{bad}}]}(\cdot) \le 2\mathbf{Adv}^{\mathrm{uce}}_{\mathsf{H},S,D}(\cdot) + 2\mathbf{Adv}^{\mathrm{ltdf}}_{\mathsf{LT},B_1}(\cdot) + \mathbf{Adv}^{\mathrm{cpa}}_{\mathsf{RE}_{\mathsf{bad}},B_2}(\cdot) + v^2/\mathsf{LT.il} \ .$$

What remains is to prove that S is statistically unpredictable. Let P be a predictor. Consider the following games H_1 and H_2. They are identical to games G_3 and G_4 respectively, except that instead of returning $(b = b')$, we'll give $(b, ek, pk', \mathbf{c})$ to P, and return 1 if and only if P can guess a message $\mathbf{m}_b[i]$ or a synthetic coin $\mathbf{r}[i]$. Game H_1 corresponds to game Pred^P_S, and

$$\Pr[H_1(\cdot) \Rightarrow 1] - \Pr[H_2(\cdot) \Rightarrow 1] \le v^2/2^{\mathsf{LT}+1} \ .$$

On the other hand, in game H_2, the predictor is only given the images of $\mathbf{r}$ under LT with a lossy key, the first ℓ bits of the messages $\mathbf{m}_b$, and some independent, random strings. The chance that it can guess some $\mathbf{r}[i]$ is at most $v/2^\tau$, and the chance that it can guess some $\mathbf{m}_b[i]$ is at most $v \cdot 2^\ell \mathrm{Guess}_A(\cdot) \le v/2^\ell$. Hence $\mathbf{Adv}^{\mathrm{pred}}_{S,P}(\cdot) \le v/2^\tau + v/2^\ell + v^2/2^{\mathsf{LT.il}}$. ▌

B Deferred Proofs

B.1 Proof of Theorem 3.2

Let $\mathsf{Rsmp}^{\mathrm{RO}}$ be an efficient δ-partial resampling algorithm of $\mathcal{M}^{\mathrm{RO}}$. Consider games G_1–G_9 in Figures 20–24. Each game maintains three independent random oracles $\mathrm{RO}, \overline{\mathrm{RO}}$, and RO^*. Procedure RO maintains a local array R as follows:

```
Procedure $\mathrm{RO}(x, \ell)$
If $R[x, \ell] = \bot$ then $R[x, \ell] \leftarrow\$ \{0,1\}^\ell$
Return $R[x, \ell]$
```

```
Game $G_2(k)$, $\boxed{G_3(k)}$
$hk \leftarrow_\$ \{0,1\}^k$ ; $done \leftarrow \mathsf{false}$ ; $coll \leftarrow \mathsf{false}$ ; $param \leftarrow_\$ A.\mathrm{pg}^{\mathrm{RO}_1}(1^k)$
$(\mathbf{m}, \mathbf{a}) \leftarrow_\$ \mathcal{M}^{\mathrm{RO}_1}(1^k, param)$ ; $z_1 \leftarrow \delta(\mathbf{m}, param)$
$ek \leftarrow_\$ \mathsf{LT.LKg}(1^k)$ ; $done \leftarrow \mathsf{true}$
For $i = 1$ to $|\mathbf{m}|$ do
    $\mathbf{r}[i] \leftarrow \mathrm{RO}(hk \,\|\, 0 \,\|\, \mathbf{m}[i], \mathsf{LT.il}(k))$ ; $trap \leftarrow \mathsf{LT.Eval}(ek, \mathbf{r}[i])$
    $y \leftarrow \mathrm{RO}(hk \,\|\, 1 \,\|\, \mathbf{r}[i], |\mathbf{m}[i]|) \oplus \mathbf{m}[i]$ ; $\mathbf{c}[i] \leftarrow (trap, y)$
$(state, I) \leftarrow_\$ A.\mathrm{cor}^{\mathrm{RO}_2}((hk, ek), \mathbf{c}, param)$ ; $\omega \leftarrow_\$ A.\mathrm{g}^{\mathrm{RO}_3}(state, \mathbf{m}[I], \mathbf{a}[I])$
$z_0 \leftarrow_\$ \mathsf{Rsmp}^{\mathrm{RO}_1}(1^k, \mathbf{m}[I], \mathbf{a}[I], I, param)$ ; $b \leftarrow_\$ \{0,1\}$ ; $t \leftarrow_\$ A.\mathrm{f}^{\mathrm{RO}_4}(z_b, param)$
If $(\omega = t)$ then return $b$ else return $1 - b$

Procedure $\mathrm{RO}_1(x, \ell)$
If $(|x| > k) \wedge (x[1,k] = hk)$ then
    If $\neg(done)$ then $\mathsf{bad} \leftarrow \mathsf{true}$ ; $\boxed{coll \leftarrow \mathsf{true}}$
    If $coll$ then return $\overline{\mathrm{RO}}(x, \ell)$
Return $\mathrm{RO}(x, \ell)$

Procedure $\mathrm{RO}_2(x, \ell)$
Return $\mathrm{RO}(x, \ell)$

Procedure $\mathrm{RO}_3(x, \ell)$
Return $\mathrm{RO}(x, \ell)$

Procedure $\mathrm{RO}_4(x, \ell)$
Return $\mathrm{RO}(x, \ell)$
```

Figure 21: **Games G_2 and G_3 of the proof of Theorem 3.2.** Game G_2 contains the corresponding boxed statement, but game G_3 does not.

For simplicity, we omit the code of $\mathrm{RO}, \overline{\mathrm{RO}}, \mathrm{RO}^*$ in the games. In each game, we use $\mathrm{RO}_1, \mathrm{RO}_2, \mathrm{RO}_3$, and RO_4 to denote the oracle interface of $(A.\mathrm{pg}, \mathcal{M}), A.\mathrm{cor}, A.\mathrm{g}$, and $A.\mathrm{f}$ respectively. Then

$$\mathbf{Adv}^{\text{d-so-cpa2}}_{\mathsf{DE1}[H,\mathsf{LT}],A,\mathcal{M}}(k) = 2\Pr[G_1(k) \Rightarrow 1] - 1 \ .$$

We now explain the game chain. Game G_2 is identical to game G_1, but instead of using an injective key generated by $\mathsf{LT.IKg}(1^k)$, we use a lossy key generated by $\mathsf{LT.LKg}(1^k)$. Game G_2 also does the following bookkeeping: (i) it maintains a variable *coll* that is always false, and (ii) in $\mathrm{RO}_1(x, \ell)$, if the prefix of x is hk and *coll* is true then we return $\overline{\mathrm{RO}}(x, \ell)$ instead of $\mathrm{RO}(x, \ell)$. Note that the code in (ii) is never executed, because *coll* is always false. Consider the following adversary $D(ek)$. It simulates game G_1, but uses its ek instead of generating an injective key. It outputs 1 if the simulated game returns 1, and outputs 0 otherwise. Then

$$\mathbf{Adv}^{\mathrm{ltdf}}_{\mathsf{LT},D}(k) = \Pr[G_1(k) \Rightarrow 1] - \Pr[G_2(k) \Rightarrow 1] \ .$$

This is the only place in the proof that we need $\mathcal{M}$ to be efficiently resamplable. Next, game G_3 is identical to game G_2, except the following: In RO_1, if hk is a prefix of some oracle queries that $A.\mathrm{pg}$ or $\mathcal{M}$ makes then we set *coll* to be true, triggering the use of $\overline{\mathrm{RO}}$. Hence in game G_3, $A.\mathrm{pg}$ and $\mathcal{M}$ know nothing about $\mathrm{RO}(hk \,\|\, x, \ell)$, for any $x \in \{0,1\}^*$ and any $\ell \in \mathbb{N}$. Note that the oracle answers of Rsmp is still always consistent with those of $\mathcal{M}$. Games G_2 and G_3 are identical-until-bad, and thus from the Fundamental Lemma of Game-playing [10],

$$\Pr[G_2(k) \Rightarrow 1] - \Pr[G_3(k) \Rightarrow 1] \le \Pr[G_2(k) \text{ sets } \mathsf{bad}] \le \frac{q(k)}{2^k} \ .$$

In game G_4, we unroll the code of procedure RO in producing the coins $\mathbf{r}[i]$. The change is conservative, meaning that $\Pr[G_4(k) \Rightarrow 1] = \Pr[G_3(k) \Rightarrow 1]$. Next, in game G_5, instead of sampling the coins $\mathbf{r}[1], \ldots, \mathbf{r}[|\mathbf{r}|]$ independently, we ensure that they are distinct. The two games G_4 and G_5 are identical-until-bad. Then from the PRP/PRF Switching Lemma,

$$\Pr[G_4(k) \Rightarrow 1] - \Pr[G_5(k) \Rightarrow 1] \le \Pr[G_4(k) \text{ sets } \mathsf{bad}] \le \frac{0.5v^2(k)}{2^{\mathsf{LT.il}(k)}} \le \frac{0.5v^2(k)}{2^{\tau(k)}} \ .$$

In game G_6, we unroll the code of procedure RO in hashing the coins $\mathbf{r}[i]$. The change is conservative. In game G_7, in procedure RO_2, if the adversary $A.\mathrm{cor}$ make a query in $\{hk \,\|\, 0 \,\|\, \mathbf{m}[i], hk \,\|\, 1 \,\|\, \mathbf{r}[i] \mid 1 \le i \le |\mathbf{m}|\}$, then the oracle lies, calling RO^* instead. Whatever $A.\mathrm{cor}$ is given is conditionally independent of $\mathbf{m}$ given *param*, and thus the chance that it can make a query in $\{hk \,\|\, 0 \,\|\, \mathbf{m}[i] \mid 1 \le i \le |\mathbf{m}|\}$ is at most $q(k)v(k)/2^{\mu(k)}$. On the other

```
Game $G_4(k)$, $\boxed{G_5(k)}$
$hk \leftarrow_\$ \{0,1\}^k$ ; $done \leftarrow \mathsf{false}$ ; $coll \leftarrow \mathsf{false}$ ; $param \leftarrow_\$ A.\mathrm{pg}^{\mathrm{RO}_1}(1^k)$
$(\mathbf{m}, \mathbf{a}) \leftarrow_\$ \mathcal{M}^{\mathrm{RO}_1}(1^k, param)$ ; $z_1 \leftarrow \delta(\mathbf{m}, param)$
$ek \leftarrow_\$ \mathsf{LT.LKg}(1^k)$ ; $done \leftarrow \mathsf{true}$
For $i = 1$ to $|\mathbf{m}|$ do
    $\mathbf{r}[i] \leftarrow_\$ \{0,1\}^{\mathsf{LT.il}(k)}$
    If $\mathbf{r}[i] \in \mathsf{Dom}$ then $\mathsf{bad} \leftarrow \mathsf{true}$ ; $\boxed{\mathbf{r}[i] \leftarrow_\$ \{0,1\}^{\mathsf{LT.il}(k)} \backslash \mathsf{Dom}}$
    $R[hk \,\|\, 0 \,\|\, \mathbf{m}[i], \mathsf{LT.il}(k)] \leftarrow \mathbf{r}[i]$
    $\mathsf{Dom} \leftarrow \mathsf{Dom} \cup \{\mathbf{r}[i]\}$ ; $trap \leftarrow \mathsf{LT.Eval}(ek, \mathbf{r}[i])$
    $y \leftarrow \mathrm{RO}(hk \,\|\, 1 \,\|\, \mathbf{r}[i], |\mathbf{m}[i]|) \oplus \mathbf{m}[i]$ ; $\mathbf{c}[i] \leftarrow (trap, y)$
$(state, I) \leftarrow_\$ A.\mathrm{cor}^{\mathrm{RO}_2}((hk, ek), \mathbf{c}, param)$ ; $\omega \leftarrow_\$ A.\mathrm{g}^{\mathrm{RO}_3}(state, \mathbf{m}[I], \mathbf{a}[I])$
$z_0 \leftarrow_\$ \mathsf{Rsmp}^{\mathrm{RO}_1}(1^k, \mathbf{m}[I], \mathbf{a}[I], I, param)$ ; $b \leftarrow_\$ \{0,1\}$ ; $t \leftarrow_\$ A.\mathrm{f}^{\mathrm{RO}_4}(z_b, param)$
If $(\omega = t)$ then return $b$ else return $1 - b$

Procedure $\mathrm{RO}_1(x, \ell)$
If $(|x| > k) \wedge (x[1,k] = hk)$ then
    If $\neg(done)$ then $coll \leftarrow \mathsf{true}$
    If $coll$ then return $\overline{\mathrm{RO}}(x, \ell)$
Return $\mathrm{RO}(x, \ell)$

Procedure $\mathrm{RO}_2(x, \ell)$
Return $\mathrm{RO}(x, \ell)$

Procedure $\mathrm{RO}_3(x, \ell)$
Return $\mathrm{RO}(x, \ell)$

Procedure $\mathrm{RO}_4(x, \ell)$
Return $\mathrm{RO}(x, \ell)$
```

Figure 22: **Games G_4 and G_5 of the proof of Theorem 3.2.** Game G_4 contains the corresponding boxed statement, but game G_5 does not.

```
Game $G_6(k)$, $\boxed{G_7(k)}$
$hk \leftarrow_\$ \{0,1\}^k$ ; $done \leftarrow \mathsf{false}$ ; $coll \leftarrow \mathsf{false}$ ; $param \leftarrow_\$ A.\mathrm{pg}^{\mathrm{RO}_1}(1^k)$
$(\mathbf{m}, \mathbf{a}) \leftarrow_\$ \mathcal{M}^{\mathrm{RO}_1}(1^k, param)$ ; $z_1 \leftarrow \delta(\mathbf{m}, param)$
$ek \leftarrow_\$ \mathsf{LT.LKg}(1^k)$ ; $done \leftarrow \mathsf{true}$ ; $\mathsf{Dom} \leftarrow \emptyset$
For $i = 1$ to $|\mathbf{m}|$ do
    $\mathbf{r}[i] \leftarrow_\$ \{0,1\}^{\mathsf{LT.il}(k)} \backslash \mathsf{Dom}$ ; $R[hk \,\|\, 0 \,\|\, \mathbf{m}[i], \mathsf{LT.il}(k)] \leftarrow \mathbf{r}[i]$
    $\mathsf{Dom} \leftarrow \mathsf{Dom} \cup \{\mathbf{r}[i]\}$ ; $trap \leftarrow \mathsf{LT.Eval}(ek, \mathbf{r}[i])$
    $y \leftarrow_\$ \{0,1\}^{|\mathbf{m}[i]|}$ ; $R[hk \,\|\, 1 \,\|\, \mathbf{r}[i], |\mathbf{m}[i]|] \leftarrow y \oplus \mathbf{m}[i]$ ; $\mathbf{c}[i] \leftarrow (trap, y)$
$V \leftarrow \{hk \,\|\, 0 \,\|\, \mathbf{m}[i], hk \,\|\, 1 \,\|\, \mathbf{r}[i] \mid 1 \le i \le |\mathbf{m}|\}$
$(state, I) \leftarrow_\$ A.\mathrm{cor}^{\mathrm{RO}_2}((hk, ek), \mathbf{c}, param)$ ; $\omega \leftarrow_\$ A.\mathrm{g}^{\mathrm{RO}_3}(state, \mathbf{m}[I], \mathbf{a}[I])$
$z_0 \leftarrow_\$ \mathsf{Rsmp}^{\mathrm{RO}_1}(1^k, \mathbf{m}[I], \mathbf{a}[I], I, param)$ ; $b \leftarrow_\$ \{0,1\}$ ; $t \leftarrow_\$ A.\mathrm{f}^{\mathrm{RO}_4}(z_b, param)$
If $(\omega = t)$ then return $b$ else return $1 - b$

Procedure $\mathrm{RO}_1(x, \ell)$
If $(|x| > k) \wedge (x[1,k] = hk)$ then
    If $\neg(done)$ then $coll \leftarrow \mathsf{true}$
    If $coll$ then return $\overline{\mathrm{RO}}(x, \ell)$
Return $\mathrm{RO}(x, \ell)$

Procedure $\mathrm{RO}_2(x, \ell)$
If $x \in V$ then $\mathsf{bad} \leftarrow \mathsf{true}$ ; $\boxed{\text{return } \mathrm{RO}^*(x, \ell)}$
Return $\mathrm{RO}(x, \ell)$

Procedure $\mathrm{RO}_3(x, \ell)$
Return $\mathrm{RO}(x, \ell)$

Procedure $\mathrm{RO}_4(x, \ell)$
Return $\mathrm{RO}(x, \ell)$
```

Figure 23: **Games G_6 and G_7 of the proof of Theorem 3.2.** Game G_6 contains the corresponding boxed statement, but game G_7 does not.

hand, since we use a lossy key in $\mathsf{LT.Eval}$ and the marginal distribution of each $\mathbf{r}[i]$ is uniformly random, the

```
Game $G_8(k)$, $\boxed{G_9(k)}$
$hk \leftarrow_\$ \{0,1\}^k$ ; $done \leftarrow \mathsf{false}$ ; $coll \leftarrow \mathsf{false}$ ; $param \leftarrow_\$ A.\mathrm{pg}^{\mathrm{RO}_1}(1^k)$
$(\mathbf{m}, \mathbf{a}) \leftarrow_\$ \mathcal{M}^{\mathrm{RO}_1}(1^k, param)$ ; $ek \leftarrow_\$ \mathsf{LT.LKg}(1^k)$ ; $done \leftarrow \mathsf{true}$ ; $\mathsf{Dom} \leftarrow \emptyset$
For $i = 1$ to $|\mathbf{m}|$ do
    $\mathbf{r}[i] \leftarrow_\$ \{0,1\}^{\mathsf{LT.il}(k)} \backslash \mathsf{Dom}$ ; $R[hk \,\|\, 0 \,\|\, \mathbf{m}[i], \mathsf{LT.il}(k)] \leftarrow \mathbf{r}[i]$
    $\mathsf{Dom} \leftarrow \mathsf{Dom} \cup \{\mathbf{r}[i]\}$ ; $trap \leftarrow \mathsf{LT.Eval}(ek, \mathbf{r}[i])$
    $y \leftarrow_\$ \{0,1\}^{|\mathbf{m}[i]|}$ ; $R[hk \,\|\, 1 \,\|\, \mathbf{r}[i], |\mathbf{m}[i]|] \leftarrow y \oplus \mathbf{m}[i]$ ; $\mathbf{c}[i] \leftarrow (trap, y)$
$V \leftarrow \{hk \,\|\, 0 \,\|\, \mathbf{m}[i], hk \,\|\, 1 \,\|\, \mathbf{r}[i] \mid 1 \le i \le |\mathbf{m}|\}$
$(state, I) \leftarrow_\$ A.\mathrm{cor}^{\mathrm{RO}_2}((hk, ek), \mathbf{c}, param)$
$L \leftarrow \{hk \,\|\, 0 \,\|\, \mathbf{m}[i], hk \,\|\, 1 \,\|\, \mathbf{m}[i] \mid i \in I\}$ ; $\omega \leftarrow_\$ A.\mathrm{g}^{\mathrm{RO}_3}(state, \mathbf{m}[I], \mathbf{a}[I])$
$z_0 \leftarrow_\$ \mathsf{Rsmp}^{\mathrm{RO}_1}(1^k, \mathbf{m}[I], \mathbf{a}[I], I, param)$ ; $b \leftarrow_\$ \{0,1\}$ ; $t \leftarrow_\$ A.\mathrm{f}^{\mathrm{RO}_4}(z_b, param)$
If $(\omega = t)$ then return $b$ else return $1 - b$
```

```
Procedure $\mathrm{RO}_1(x, \ell)$
If $(|x| > k) \wedge (x[1,k] = hk)$ then
    If $\neg(done)$ then $coll \leftarrow \mathsf{true}$
    If $coll$ then return $\overline{\mathrm{RO}}(x, \ell)$
Return $\mathrm{RO}(x, \ell)$

Procedure $\mathrm{RO}_2(x, \ell)$
If $x \in V$ then return $\mathrm{RO}^*(x, \ell)$
Return $\mathrm{RO}(x, \ell)$

Procedure $\mathrm{RO}_3(x, \ell)$
If $x \in V \backslash L$ then
    $\mathsf{bad} \leftarrow \mathsf{true}$ ; $\boxed{\text{return } \mathrm{RO}^*(x, \ell)}$
Return $\mathrm{RO}(x, \ell)$

Procedure $\mathrm{RO}_4(x, \ell)$
Return $\mathrm{RO}(x, \ell)$
```

Figure 24: **Games G_8 and G_9 of the proof of Theorem 3.2.** Game G_8 contains the corresponding boxed statement, but game G_9 does not.

chance that A.cor can make a query in $\{hk \,\|\, 1 \,\|\, \mathbf{r}[i] \mid 1 \le i \le |\mathbf{m}|\}$ is at most $q(k)v(k))/2^{\tau(k)}$. Hence

$$\begin{aligned}\Pr[G_6(k) \Rightarrow 1] - \Pr[G_7(k) \Rightarrow 1] &\le \Pr[G_7(k) \text{ sets } \mathsf{bad}] \\ &\le \frac{q(k)v(k)}{2^{\mu(k)}} + \frac{q(k)v(k)}{2^{\tau(k)}} \ .\end{aligned}$$

Game G_8 is a simplified version of game G_7. In game G_9, if A.g makes a query in $\{hk \,\|\, 0 \,\|\, \mathbf{m}[i], hk \,\|\, 1 \,\|\, \mathbf{r}[i] \mid i \notin I\}$, to RO_3, then the oracle lies, calling RO^* instead. Since the ciphertexts are independent of $\mathbf{m}$, and $\mathcal{M}$ is (μ, d)-entropic, and A.cor opens at most d ciphertexts, the chance that A.g can make a query in $\{hk \,\|\, 0 \,\|\, \mathbf{m}[i], \mid i \notin I\}$, is at most $q(k)v(k)/2^{\mu(k)}$. Let X_j be the random variable of the number of preimages of $\mathsf{LT.Eval}(ek, \mathbf{r}[j])$. Then the chance that A.g can make a query $\{hk \,\|\, 1 \,\|\, \mathbf{r}[i] \mid i \notin I\}$ is at most

$$q(k) \cdot \mathbf{E}\Big(\sum_{i \notin I} \frac{1}{X_i}\Big) \le q(k) \cdot \mathbf{E}\Big(\sum_{i=1}^{v(k)} \frac{1}{X_i}\Big) \le q(k) \cdot \sum_{i=1}^{v(k)} \mathbf{E}\Big(\frac{1}{X_i}\Big) \ .$$

We claim that for every $j \in \{1, \ldots, v(k)\}$, we have

$$\mathbf{E}\Big(\frac{1}{X_j}\Big) \le 2^{-\tau(k)} \tag{1}$$

and thus

$$\Pr[G_8(k) \Rightarrow 1] - \Pr[G_9(k) \Rightarrow 1] \le \frac{q(k)v(k)}{2^{\mu(k)}} + \frac{q(k)v(k)}{2^{\tau(k)}} \ .$$

To justify Equation (1), fix $j \in \{1, \ldots, v(k)\}$ and let $S = \{\mathsf{LT.Eval}(ek, r) \mid r \in \{0,1\}^{\mathsf{LT.il}(k)}\}$. For each $y \in S$, let N_y be the number of strings $r \in \{0,1\}^{\mathsf{LT.il}(k)}$ such that $\mathsf{LT.Eval}(ek, r) = y$. Since $\Pr[\mathsf{LT.Eval}(ek, \mathbf{r}[j]) = y] = N_y/2^{\mathsf{LT.il}(k)}$,

$$\mathbf{E}\Big(\frac{1}{X_j}\Big) = \sum_{y \in S} \frac{N_y}{2^{\mathsf{LT.il}(k)}} \cdot \frac{1}{N_y} = \frac{|S|}{2^{\mathsf{LT.il}(k)}} \le 2^{-\tau(k)}$$

as claimed.

Next, note that in game G_9, whatever A.cor receives is independent of $\mathbf{m}$, so the set I is conditionally independent of $\mathbf{m}$, given *param*. Hence conditioning on *param*, the distribution of the resampled string z_0 is the same as $z_1 = \delta(\mathbf{m}, \mathit{param})$, and both are conditionally independent of hk given *param*. Let Bad be the event that A.f can make a query whose prefix is hk, which happens with probability at most $q(k)/2^k$. If Bad doesn't happen then from the joint view of A.f and A.g, the string z_0 is identically distributed as z_1, and the output of A.f is conditionally independent of the ciphertexts and the public key, given *param*. Hence $\Pr[G_9(\cdot) \Rightarrow 1] \le 1/2 + q(k)/2^k$. Summing up,

$$\begin{aligned}\mathbf{Adv}^{\text{d-so-cpa2}}_{\mathsf{DE1}[H,\mathsf{LT}],A,\mathcal{M}}(k) &= 2\Pr[G_1(k) \Rightarrow 1] - 1 \\ &\le \frac{4q(k)}{2^k} + \frac{v(k)(v(k)+4q(k))}{2^{\tau(k)}} + \frac{4q(k)v(k)}{2^{\mu(k)}} + 2\mathbf{Adv}^{\text{ltdf}}_{\mathsf{LT},D}(k)\end{aligned}$$

as claimed.

B.2 Proof of Theorem 4.1

Let Rsmp be an efficient δ-partial resampling algorithm of $\mathcal{M}$. Consider games G_1–G_{10} in Figure 25, Figure 26, and Figure 27. The games maintain a procedure RO as follows.

```
**Procedure** RO($x, \ell$)
If $R[x,\ell] = \bot$ then $R[x,\ell] \leftarrow_{\$} \{0,1\}^{\ell}$
Return $R[x,\ell]$
```

Let RO_1 denote the interface to the random oracle of $\mathcal{M}$ and Rsmp, and also A's interface to the random oracle before it's given the ciphertexts. Let RO_2 denote A's interface to the random oracle after it's given the ciphertexts, but still not given the messages, and let RO_3 denote A's interface to the random oracle after it's given the messages. Wlog, assume that A doesn't repeat a prior random-oracle query, and assume further that $q(k) \le 2^k$. We now explain the game chain. Game G_1 corresponds to game $\text{N-SO-CPA}^{A,\mathcal{M},\delta}_{\mathsf{NE1}[H,\mathsf{LT}],\mathsf{NG}}$. Game G_2 is identical to game G_1, except that the seeds $\boldsymbol{xk}[j]$ are distinct. Then from the PRP/PRF Switching Lemma

$$\Pr[G_1(k) \Rightarrow 1] - \Pr[G_2(k) \Rightarrow 1] \le \frac{v^2(k)}{2^{k+1}} \ .$$

Next, in game G_3, instead of using an injective key for LT, we'll use a lossy key. Consider the following adversary B against LT. It simulates game G_3, but uses the given key instead of running $\mathsf{LT.LKg}$, and outputs 1 if and only if the simulated game returns 1. Then

$$\Pr[G_2(\cdot) \Rightarrow 1] - \Pr[G_3(\cdot) \Rightarrow 1] \le \mathbf{Adv}^{\text{ltdf}}_{\mathsf{LT},B}(\cdot) \ .$$

Next, game G_4 is identical to game G_3, but for each sender $j \in J$, if some of its nonces (generated by NG) repeats then we prematurely terminate the game and return 0, and thus effectively let the adversary lose. Consider the following adversary D attacking NG. It runs A and simulates game G_3, but with the following changes. When D has to enter the loop to generate the ciphertexts, it picks $j \leftarrow_{\$} J$. For sender j, instead of running NG to generate the nonces, D will use the nonce selectors of sender j to feed its oracle RP. Since D doesn't know the nonces of sender j, it has to pick the coins $\mathbf{r}[i]$ for sender j uniformly at random, instead of calling RO. The adversary D also records random-oracle queries of A and $\mathcal{M}$ of the form $(hk \,\|\, 0 \,\|\, (\boldsymbol{xk}[j], N, m), \mathsf{LT.il}(k))$, and stores the components N in a set S. Finally it outputs a random element in S. Then

$$\begin{aligned}\Pr[G_3(\cdot) \Rightarrow 1] - \Pr[G_4(\cdot) \Rightarrow 1] &\le \Pr[G_3(\cdot) \text{ sets } \mathsf{bad}] \\ &\le v \cdot \mathbf{Adv}^{\text{rp}}_{\mathsf{NG},D}(\cdot) \le qv \cdot \mathbf{Adv}^{\text{rp}}_{\mathsf{NG},D}(\cdot) \ .\end{aligned}$$

Game G_5 is identical to game G_4, plus some bookkeeping. Hence,

$$\Pr[G_5(\cdot) \Rightarrow 1] = \Pr[G_4(\cdot) \Rightarrow 1] \ .$$

```
Games $G_1(k)$, $G_2(k)$
$(ek, td) \leftarrow_{\$} \mathsf{LT.IKg}(1^k)$ ; $\mathsf{Dom} \leftarrow \emptyset$
$hk \leftarrow_{\$} \{0,1\}^k$ ; $pk \leftarrow (hk, ek)$
$(J, state) \leftarrow_{\$} A^{\mathrm{RO}_1}(1^k, pk)$
For $j = 1, \ldots, v(k)$ do
    $\boldsymbol{xk}[j] \leftarrow_{\$} \{0,1\}^k$ ;  $\boldsymbol{st}[j] \leftarrow \varepsilon$
    If $\boldsymbol{xk}[j] \in \mathsf{Dom}$ then
        $\mathsf{bad} \leftarrow \mathsf{true}$ ;  $\boldsymbol{xk}[j] \leftarrow_{\$} \{0,1\}^k \backslash \mathsf{Dom}$
    $\mathsf{Dom} \leftarrow \mathsf{Dom} \cup \{\boldsymbol{xk}[j]\}$
$(param, \boldsymbol{N}, U, \boldsymbol{\sigma}, state) \leftarrow_{\$} A^{\mathrm{RO}_1}(state, \boldsymbol{xk}[J])$
$(\mathbf{m}, \mathbf{a}) \leftarrow_{\$} \mathcal{M}^{\mathrm{RO}_1}(1^k, param)$ ; $b \leftarrow_{\$} \{0,1\}$
For $i = 1$ to $|\mathbf{m}|$ do
    $j \leftarrow U[i]$
    If $j \in J$ then
        $(N, \boldsymbol{st}[j]) \leftarrow_{\$} \mathsf{NG}(1^k, \boldsymbol{st}[j], \boldsymbol{\sigma}[i])$
        $\boldsymbol{N}[i] \leftarrow N$
    $x \leftarrow (\boldsymbol{xk}[j], \boldsymbol{N}[i], \mathbf{m}[i])$
    $\mathbf{r}[i] \leftarrow \mathrm{RO}(hk \,\|\, 0 \,\|\, x, \mathsf{LT.il}(k))$
    $y \leftarrow \mathrm{RO}(hk \,\|\, 1 \,\|\, \mathbf{r}[i], |\mathbf{m}[i]|) \oplus \mathbf{m}[i]$
    $trap \leftarrow \mathsf{LT.Eval}(ek, \mathbf{r}[i])$
    $\mathbf{c}[i] \leftarrow (trap, y)$
$(I, state) \leftarrow_{\$} A^{\mathrm{RO}_2}(state, \mathbf{c})$
For $i \in I$, $j = 1$ to $|\mathbf{m}|$ do
    If $U[i] = U[j]$ then $I \leftarrow I \cup \{j\}$
$z_0 \leftarrow_{\$} \mathsf{Rsmp}^{\mathrm{RO}_1}(1^k, \mathbf{m}[I], \mathbf{a}[I], I, param)$
$z_1 \leftarrow \delta(\mathbf{m}, param)$
$b' \leftarrow_{\$} A^{\mathrm{RO}_3}(state, \mathbf{m}[I], \mathbf{a}[I], \boldsymbol{N}[I], \boldsymbol{xk}[U[I]], z_b)$
Return $(b = b')$
```

```
Games $G_3(k)$, $G_4(k)$
$ek \leftarrow_{\$} \mathsf{LT.LKg}(1^k)$ ; $\mathsf{Dom} \leftarrow \emptyset$
$hk \leftarrow_{\$} \{0,1\}^k$ ; $pk \leftarrow (hk, ek)$
$(J, state) \leftarrow_{\$} A^{\mathrm{RO}_1}(1^k, pk)$
For $j = 1, \ldots, v(k)$ do
    $\boldsymbol{xk}[j] \leftarrow_{\$} \{0,1\}^k \backslash \mathsf{Dom}$ ;  $\boldsymbol{st}[j] \leftarrow \varepsilon$
    $\mathsf{Dom} \leftarrow \mathsf{Dom} \cup \{\boldsymbol{xk}[j]\}$ ; $S_j \leftarrow \emptyset$
$(param, \boldsymbol{N}, U, \boldsymbol{\sigma}, state) \leftarrow_{\$} A^{\mathrm{RO}_1}(state, \boldsymbol{xk}[J])$
$(\mathbf{m}, \mathbf{a}) \leftarrow_{\$} \mathcal{M}^{\mathrm{RO}_1}(1^k, param)$ ; $b \leftarrow_{\$} \{0,1\}$
For $i = 1$ to $|\mathbf{m}|$ do
    $j \leftarrow U[i]$
    If $j \in J$ then
        $(N, \boldsymbol{st}[j]) \leftarrow_{\$} \mathsf{NG}(1^k, \boldsymbol{st}[j], \boldsymbol{\sigma}[i])$
        If $N \in S_j$ then
            $\mathsf{bad} \leftarrow \mathsf{true}$ ;  Return 0
        $\boldsymbol{N}[i] \leftarrow N$ ; $S_j \leftarrow S_j \cup \{N\}$
    $x \leftarrow (\boldsymbol{xk}[j], \boldsymbol{N}[i], \mathbf{m}[i])$
    $\mathbf{r}[i] \leftarrow \mathrm{RO}(hk \,\|\, 0 \,\|\, x, \mathsf{LT.il}(k))$
    $y \leftarrow \mathrm{RO}(hk \,\|\, 1 \,\|\, \mathbf{r}[i], |\mathbf{m}[i]|) \oplus \mathbf{m}[i]$
    $trap \leftarrow \mathsf{LT.Eval}(ek, \mathbf{r}[i])$
    $\mathbf{c}[i] \leftarrow (trap, y)$
$(I, state) \leftarrow_{\$} A^{\mathrm{RO}_2}(state, \mathbf{c})$
For $i \in I$, $j = 1$ to $|\mathbf{m}|$ do
    If $U[i] = U[j]$ then $I \leftarrow I \cup \{j\}$
$z_0 \leftarrow_{\$} \mathsf{Rsmp}^{\mathrm{RO}_1}(1^k, \mathbf{m}[I], \mathbf{a}[I], I, param)$
$z_1 \leftarrow \delta(\mathbf{m}, param)$
$b' \leftarrow_{\$} A^{\mathrm{RO}_3}(state, \mathbf{m}[I], \mathbf{a}[I], \boldsymbol{N}[I], \boldsymbol{xk}[U[I]], z_b)$
Return $(b = b')$
```

```
Procedure $\mathrm{RO}_1(x, \ell)$
If $H[x, \ell] = \perp$ then $H[x, \ell] \leftarrow \mathrm{RO}(x, \ell)$
Return $H[x, \ell]$
```

```
Procedure $\mathrm{RO}_2(x, \ell)$
Return $\mathrm{RO}(x, \ell)$

Procedure $\mathrm{RO}_3(x, \ell)$
Return $\mathrm{RO}(x, \ell)$
```

Figure 25: **Games G_1–G_4 in the proof of Theorem 4.1.** The games share the same code for procedures $\mathrm{RO}_1, \mathrm{RO}_2, \mathrm{RO}_3$, and RO_4.

Next, in game G_6, recall that to generate the ciphertexts, for every $i \le v(k)$, one has to call

$$\mathbf{r}[i] \leftarrow \mathrm{RO}(hk \,\|\, 0 \,\|\, (\boldsymbol{xk}[U[i]], \boldsymbol{N}[i], \mathbf{m}[i]), \mathsf{LT.il}(k)) \ .$$

If there's some prior query $\mathrm{RO}_1(hk \,\|\, 0 \,\|\, (\boldsymbol{xk}[U[i]], \boldsymbol{N}[i], m), \mathsf{LT.il}(k))$, instead of using the consistent answer, we picks a random value for $\mathbf{r}[i]$ and overwrite RO to make $\mathbf{r}[i]$ consistent with RO. Note that now RO and RO_1 are no longer consistent, but (i) RO_2 and RO_3 are still consistent with RO, and (ii) subsequent queries to RO_1 by Rsmp will be consistent with prior queries by $\mathcal{M}$, instead of using the overwritten entries in RO. Then

$$\Pr[G_5(\cdot) \Rightarrow 1] - \Pr[G_6(\cdot) \Rightarrow 1] \le \Pr[G_5(\cdot) \text{ sets } \mathsf{bad}] \ .$$

We now bound the chance that game $G_5(k)$ sets bad. Let i be the number that, computing the ciphertext for $\mathbf{m}[i]$ triggers G_4 to set bad, and let j be the sender of $\mathbf{m}[i]$. If $j \notin J$, meaning that its seed $\boldsymbol{xk}[j]$ is secret, then the chance that G_5 sets bad is at most $v(k)q(k)/(2^k - v(k)) \le v(k)(q(k) + v(k))/2^k$, because marginally, each $\boldsymbol{xk}[j]$ is uniformly distributed over a set of at least $2^k - v(k)$ elements. If $j \in J$ then the probability that game

```
**Games** $G_5(k)$, $G_6(k)$
$ek \leftarrow\$ \mathsf{LT.LKg}(1^k)$ ; $\mathsf{Dom}, V_0, V_1 \leftarrow \emptyset$
$hk \leftarrow\$ \{0,1\}^k$ ; $pk \leftarrow (hk, ek)$
$(J, state) \leftarrow\$ A^{\mathrm{RO}_1}(1^k, pk)$
For $j = 1, \dots, v(k)$ do
    $\boldsymbol{xk}[j] \leftarrow\$ \{0,1\}^k \backslash \mathsf{Dom}$ ; $\boldsymbol{st}[j] \leftarrow \varepsilon$
    $\mathsf{Dom} \leftarrow \mathsf{Dom} \cup \{\boldsymbol{xk}[j]\}$ ; $S_j \leftarrow \emptyset$
$(param, \boldsymbol{N}, U, \boldsymbol{\sigma}, state) \leftarrow\$ A^{\mathrm{RO}_1}(state, \boldsymbol{xk}[J])$
$(\mathbf{m}, \mathbf{a}) \leftarrow\$ \mathcal{M}^{\mathrm{RO}_1}(1^k, param)$ ; $b \leftarrow\$ \{0,1\}$
For $i = 1$ to $|\mathbf{m}|$ do
    $j \leftarrow U[i]$
    If $j \in J$ then
        $(N, \boldsymbol{st}[j]) \leftarrow\$ \mathsf{NG}(1^k, \boldsymbol{st}[j], \boldsymbol{\sigma}[i])$
        If $N \in S_j$ then return 0
        $\boldsymbol{N}[i] \leftarrow N$ ; $S_j \leftarrow S_j \cup \{N\}$
    $x \leftarrow (\boldsymbol{xk}[j], \boldsymbol{N}[i], \mathbf{m}[i])$
    If $(\boldsymbol{xk}[j], \boldsymbol{N}[i]) \in V_0$ then
        $\mathsf{bad} \leftarrow \mathsf{true}$
        $R[hk \,\|\, 0 \,\|\, x, \mathsf{LT.il}(k)] \leftarrow\$ \{0,1\}^{\mathsf{LT.il}(k)}$
    $\mathbf{r}[i] \leftarrow \mathrm{RO}(hk \,\|\, 0 \,\|\, x, \mathsf{LT.il}(k))$
    $y \leftarrow \mathrm{RO}(hk \,\|\, 1 \,\|\, \mathbf{r}[i], |\mathbf{m}[i]|) \oplus \mathbf{m}[i]$
    $trap \leftarrow \mathsf{LT.Eval}(ek, \mathbf{r}[i])$
    $\mathbf{c}[i] \leftarrow (trap, y)$
$(I, state) \leftarrow\$ A^{\mathrm{RO}_2}(state, \mathbf{c})$
For $i \in I$, $j = 1$ to $|\mathbf{m}|$ do
    If $U[i] = U[j]$ then $I \leftarrow I \cup \{j\}$
$z_0 \leftarrow\$ \mathsf{Rsmp}^{\mathrm{RO}_1}(1^k, \mathbf{m}[I], \mathbf{a}[I], I, param)$
$z_1 \leftarrow \delta(\mathbf{m}, param)$
$b' \leftarrow\$ A^{\mathrm{RO}_3}(state, \mathbf{m}[I], \mathbf{a}[I], \boldsymbol{N}[I], \boldsymbol{xk}[U[I]], z_b)$
Return $(b = b')$
```

```
**Games** $G_7(k)$, $G_8(k)$
$ek \leftarrow\$ \mathsf{LT.LKg}(1^k)$ ; $\mathsf{Dom}, V_0, V_1 \leftarrow \emptyset$
$hk \leftarrow\$ \{0,1\}^k$ ; $pk \leftarrow (hk, ek)$
$(J, state) \leftarrow\$ A^{\mathrm{RO}_1}(1^k, pk)$
For $j = 1, \dots, v(k)$ do
    $\boldsymbol{xk}[j] \leftarrow\$ \{0,1\}^k \backslash \mathsf{Dom}$ ; $\boldsymbol{st}[j] \leftarrow \varepsilon$
    $\mathsf{Dom} \leftarrow \mathsf{Dom} \cup \{\boldsymbol{xk}[j]\}$ ; $S_j \leftarrow \emptyset$
$(param, \boldsymbol{N}, U, \boldsymbol{\sigma}, state) \leftarrow\$ A^{\mathrm{RO}_1}(state, \boldsymbol{xk}[J])$
$(\mathbf{m}, \mathbf{a}) \leftarrow\$ \mathcal{M}^{\mathrm{RO}_1}(1^k, param)$ ; $b \leftarrow\$ \{0,1\}$
For $i = 1$ to $|\mathbf{m}|$ do
    $j \leftarrow U[i]$
    If $j \in J$ then
        $(N, \boldsymbol{st}[j]) \leftarrow\$ \mathsf{NG}(1^k, \boldsymbol{st}[j], \boldsymbol{\sigma}[i])$
        If $N \in S_j$ then return 0
        $\boldsymbol{N}[i] \leftarrow N$ ; $S_j \leftarrow S_j \cup \{N\}$
    $x \leftarrow (\boldsymbol{xk}[j], \boldsymbol{N}[i], \mathbf{m}[i])$
    $\mathbf{r}[i] \leftarrow\$ \{0,1\}^{\mathsf{LT.il}(k)}$
    If $\mathbf{r}[i] \in V_1$ then
        $\mathsf{bad} \leftarrow \mathsf{true}$ ; $\mathbf{r}[i] \leftarrow\$ \{0,1\}^{\mathsf{LT.il}(k)} \backslash V_1$
        $V_1 \leftarrow V_1 \cup \{\mathbf{r}[i]\}$
    $R[hk \,\|\, 0 \,\|\, x, \mathsf{LT.il}(k)] \leftarrow \mathbf{r}[i]$
    $y \leftarrow \mathrm{RO}(hk \,\|\, 1 \,\|\, \mathbf{r}[i], |\mathbf{m}[i]|) \oplus \mathbf{m}[i]$
    $trap \leftarrow \mathsf{LT.Eval}(ek, \mathbf{r}[i])$
    $\mathbf{c}[i] \leftarrow (trap, y)$
$(I, state) \leftarrow\$ A^{\mathrm{RO}_2}(state, \mathbf{c})$
For $i \in I$, $j = 1$ to $|\mathbf{m}|$ do
    If $U[i] = U[j]$ then $I \leftarrow I \cup \{j\}$
$z_0 \leftarrow\$ \mathsf{Rsmp}^{\mathrm{RO}_1}(1^k, \mathbf{m}[I], \mathbf{a}[I], I, param)$
$z_1 \leftarrow \delta(\mathbf{m}, param)$
$b' \leftarrow\$ A^{\mathrm{RO}_3}(state, \mathbf{m}[I], \mathbf{a}[I], \boldsymbol{N}[I], \boldsymbol{xk}[U[I]], z_b)$
Return $(b = b')$
```

```
**Procedure** $\mathrm{RO}_1(x, \ell)$
If $x[1] = hk \,\|\, 1 \,\|\, s$ then $V_1 \leftarrow V_1 \cup \{s\}$
If $x = hk \,\|\, 0 \,\|\, (xk, N, m)$ then $V_0 \leftarrow (xk, N)$
If $H[x, \ell] = \bot$ then $H[x, \ell] \leftarrow \mathrm{RO}(x, \ell)$
Return $H[x, \ell]$
```

```
**Procedure** $\mathrm{RO}_2(x, \ell)$
Return $\mathrm{RO}(x, \ell)$

**Procedure** $\mathrm{RO}_3(x, \ell)$
Return $\mathrm{RO}(x, \ell)$
```

Figure 26: **Games G_5–G_8 in the proof of Theorem 4.1.** The games share the same code for procedures $\mathrm{RO}_1, \mathrm{RO}_2, \mathrm{RO}_3$, and RO_4.

G_5 sets bad is at most $q(k)v(k) \cdot \mathbf{Adv}^{\mathrm{rp}}_{\mathsf{NG},D}(k)$. Hence,

$$\Pr[G_5(k) \Rightarrow 1] - \Pr[G_6(k) \Rightarrow 1] \leq \frac{v(k)(q(k)+v(k))}{2^k} + q(k)v(k) \cdot \mathbf{Adv}^{\mathrm{rp}}_{\mathsf{NG},D}(k) \ .$$

In game G_7, instead of first running RO to pick a uniformly random value and then assigning that to the coins $\mathbf{r}[i]$, we'll choose $\mathbf{r}[i]$ uniformly at random, and set RO to be consistent with $\mathbf{r}[i]$. Then $\Pr[G_7(\cdot) \Rightarrow 1] = \Pr[G_6(\cdot) \Rightarrow 1]$. In game G_8, instead of sampling $\mathbf{r}[1], \dots, \mathbf{r}[v(k)]$ uniformly, we'll pick them so that (i) they are distinct, and (ii) there's no prior query $\mathrm{RO}_1(1 \,\|\, \mathbf{r}[i], \ell)$, for any $i \in \{1, \dots, v(k)\}$ and any $\ell \in \mathbb{N}$. Then

$$\Pr[G_7(\cdot) \Rightarrow 1] - \Pr[G_8(\cdot) \Rightarrow 1] \leq \frac{v(v+q)}{2^{\mathsf{LT.il}}} \leq \frac{v(v+q)}{2^{\tau}} \ .$$

```
**Games** $G_9(k)$, $\boxed{G_{10}(k)}$
$ek \leftarrow_{\$} \mathsf{LT.LKg}(1^k)$ ; $\mathsf{Dom}, V_0, V_1 \leftarrow \emptyset$
$hk \leftarrow_{\$} \{0,1\}^k$ ; $pk \leftarrow (hk, ek)$ ; $(J, state) \leftarrow_{\$} A^{\mathrm{RO}_1}(1^k, pk)$
For $j = 1, \ldots, v(k)$ do
    $\boldsymbol{xk}[j] \leftarrow_{\$} \{0,1\}^k \backslash \mathsf{Dom}$ ; $\boldsymbol{st}[j] \leftarrow \varepsilon$ ; $\mathsf{Dom} \leftarrow \mathsf{Dom} \cup \{\boldsymbol{xk}[j]\}$ ; $S_j \leftarrow \emptyset$
$(param, \boldsymbol{N}, U, \boldsymbol{\sigma}, state) \leftarrow_{\$} A^{\mathrm{RO}_1}(state, \boldsymbol{xk}[J])$
$(\mathbf{m}, \mathbf{a}) \leftarrow_{\$} \mathcal{M}^{\mathrm{RO}_1}(1^k, param)$ ; $b \leftarrow_{\$} \{0,1\}$
For $i = 1$ to $|\mathbf{m}|$ do
    $j \leftarrow U[i]$
    If $j \in J$ then
        $(N, \boldsymbol{st}[j]) \leftarrow_{\$} \mathsf{NG}(1^k, \boldsymbol{st}[j], \boldsymbol{\sigma}[i])$ ; If $N \in S_j$ then return 0
        $\boldsymbol{N}[i] \leftarrow N$ ; $S_j \leftarrow S_j \cup \{N\}$
    $x \leftarrow (\boldsymbol{xk}[j], \boldsymbol{N}[i], \mathbf{m}[i])$ ; $\mathbf{r}[i] \leftarrow_{\$} \{0,1\}^{\mathsf{LT.il}(k)} \backslash V_1$ ; $V_1 \leftarrow V_1 \cup \{\mathbf{r}[i]\}$
    $R[hk \,\|\, 0 \,\|\, x, \mathsf{LT.il}(k)] \leftarrow \mathbf{r}[i]$ ; $y \leftarrow \mathrm{RO}(hk \,\|\, 1 \,\|\, \mathbf{r}[i], |\mathbf{m}[i]|) \oplus \mathbf{m}[i]$
    $trap \leftarrow \mathsf{LT.Eval}(ek, \mathbf{r}[i])$ ; $\mathbf{c}[i] \leftarrow (trap, y)$
$W \leftarrow \{hk \,\|\, 0 \,\|\, (\boldsymbol{xk}[U[i]], \boldsymbol{N}[i], m), hk \,\|\, 1 \,\|\, \mathbf{r}[i] \mid 1 \le i \le v(k), m \in \{0,1\}^*\}$
$(I, state) \leftarrow_{\$} A^{\mathrm{RO}_2}(state, \mathbf{c})$
For $i \in I$, $j = 1$ to $|\mathbf{m}|$ do
    If $U[i] = U[j]$ then $I \leftarrow I \cup \{j\}$
$z_0 \leftarrow_{\$} \mathsf{Rsmp}^{\mathrm{RO}_1}(1^k, \mathbf{m}[I], \mathbf{a}[I], I, param)$ ; $z_1 \leftarrow \delta(\mathbf{m}, param)$
$b' \leftarrow_{\$} A^{\mathrm{RO}_3}(state, \mathbf{m}[I], \mathbf{a}[I], \boldsymbol{N}[I], \boldsymbol{xk}[U[I]], z_b)$
Return $(b = b')$
```

```
**Procedure** $\mathrm{RO}_1(x, \ell)$
If $x[1] = hk \,\|\, 1 \,\|\, s$ then $V_1 \leftarrow V_1 \cup \{s\}$
If $x = hk \,\|\, 0 \,\|\, (xk, N, m)$ then
    $V_0 \leftarrow (xk, N)$
If $H[x, \ell] = \bot$ then $H[x, \ell] \leftarrow \mathrm{RO}(x, \ell)$
Return $H[x, \ell]$

**Procedure** $\mathrm{RO}_2(x, \ell)$
If $x \in W$ then
    $\mathsf{bad} \leftarrow \mathsf{true}$ ; $y \leftarrow_{\$} \{0,1\}^\ell$ ; $\boxed{\text{Return } y}$
Return $\mathrm{RO}(x, \ell)$

**Procedure** $\mathrm{RO}_3(x, \ell)$
Return $\mathrm{RO}(x, \ell)$
```

Figure 27: **Games G_9 and G_{10} in the proof of Theorem 4.1.** The games share the same code for procedures $\mathrm{RO}_1, \mathrm{RO}_2, \mathrm{RO}_3$, and RO_4.

Game G_9 is the simplified version of game G_8, and thus

$$\Pr[G_9(\cdot) \Rightarrow 1] = \Pr[G_8(\cdot) \Rightarrow 1] \ .$$

Next, game G_{10} is identical to game G_9, except that for queries (x, ℓ) to RO_2, if $x \in \{hk \,\|\, 0 \,\|\, (\boldsymbol{xk}[U[i]], \boldsymbol{N}[i], m), hk \,\|\, 1 \,\|\, \mathbf{r}[i] \mid 1 \le i \le v(k), m \in \{0,1\}^*\}$ then the oracle lies, giving a random answer, independent of RO. Note that now the marginal distribution of each $\mathbf{r}[i]$ is not uniformly random anymore, but it's still statistically $q/2^{\mathsf{LT.il}} \le q/2^\tau$ close to uniform. Moreover, the adversary is given the image of $\mathbf{r}[i]$ under $\mathsf{LT.Eval}$ with a lossy key. Hence the chance that there's a query $\mathrm{RO}_2(x, \ell)$ such that $x \in \{hk \,\|\, 1 \,\|\, \mathbf{r}[i] \mid 1 \le i \le v(k)\}$ is at most $2qv/2^\tau$. On the other hand, the chance that there's a query $\mathrm{RO}_2(x, \ell)$ such that $x \in \{hk \,\|\, 0 \,\|\, (\boldsymbol{xk}[U[i]], \boldsymbol{N}[i], m) \mid 1 \le i \le v(k), m \in \{0,1\}^*, U[i] \in J\}$ is at most

$$qv \cdot \mathbf{Adv}^{\mathrm{rp}}_{\mathsf{NG},D}(\cdot) + \frac{v^2}{2^{\mathsf{LT.il}+1}} \le qv \cdot \mathbf{Adv}^{\mathrm{rp}}_{\mathsf{NG},D}(\cdot) + \frac{v^2}{2^{\tau+1}},$$

where the term $v^2/2^{\mathsf{LT.il}+1}$ accounts for the failure of D to produce distinct $\mathbf{r}[1], \ldots, \mathbf{r}[v(k)]$.[10] Moreover, the chance that there's a query $\mathrm{RO}_2(x, \ell)$ such that $x \in \{hk \,\|\, 0 \,\|\, (\boldsymbol{xk}[U[i]], \boldsymbol{N}[i], m) \mid 1 \le i \le v(k), m \in \{0,1\}^*, U[i] \notin J\}$ is

[10]Note that in the simulated game by D, the game doesn't terminate when NG generates a repeated nonce, but in that case, the adversary A is given advantage 0 anyway. The simulated game also behaves different from game G_9 if the adversary A can query some $(0 \,\|\, (xk, N, m), \ell)$ such that N is a nonce generated by NG, but then D can add this N to its set S.

```
Procedure Dec(c)
$(trap, y, z) \leftarrow c$
For $r \in S$ do
    If $(\mathsf{LT.Eval}(ek, r) = trap)$ then
        $m \leftarrow \mathrm{RO}(hk \,\|\, 01 \,\|\, r, |y|) \oplus y$
        If $(z = \mathrm{RO}(hk \,\|\, 10 \,\|\, r \,\|\, m, k))$ then return $m$
Return $\perp$
```

```
Procedure $\overline{\mathrm{RO}}(x, \ell)$
If $(\ell = k)$ then
    If $(x = hk \,\|\, 10 \,\|\, w)$ then
        $S \leftarrow S \cup \{w[1, \mathsf{LT.il}(k)]\}$
Return $\mathrm{RO}(x, \ell)$
```

Figure 28: **Procedures $\overline{\mathrm{RO}}$ and Dec in game G_2 in the proof of Theorem 4.2.** The set S is initialized to be the empty set.

at most $v(k)(q(k)+v(k))/2^k$. Then

$$\Pr[G_9(k) \Rightarrow 1] - \Pr[G_{10}(k) \Rightarrow 1] \le \frac{v(k)(q(k)+v(k))}{2^k} + q(k)v(k) \cdot \mathbf{Adv}^{\mathrm{rp}}_{\mathsf{NG},D}(k) + \frac{0.5v^2(k) + 2q(k)v(k)}{2^{\tau(k)}} .$$

In game G_{10}, let Bad be the event that there's some query (x,ℓ) to RO_3 such that $x \in \{hk \,\|\, 0 \,\|\, (\boldsymbol{xk}[U[i]], \boldsymbol{N}[i], m) \mid 1 \,\|\, \mathbf{r}[i] \mid i \notin I, m \in \{0,1\}^*\}$. The chance that there's a query $\mathrm{RO}_3(x,\ell)$ such that $x \in \{hk \,\|\, 0 \,\|\, (\boldsymbol{xk}[U[i]], \boldsymbol{N}[i], m) \mid i \notin I, m \in \{0,1\}^*\}$ is at most

$$\frac{v(k)(q(k)+v(k))}{2^k} + q(k)v(k) \cdot \mathbf{Adv}^{\mathrm{rp}}_{\mathsf{NG},D}(k) + \frac{v^2(k)}{2^{\tau(k)+1}} .$$

Next, if the marginal distribution of each $\mathbf{r}[i]$ *were* uniform (which is not true), then similar to the proof of Theorem 3.2, the chance that one can query $(1 \,\|\, \mathbf{r}[i], \ell)$ for some $i \notin I$, after adaptively choosing I, is at most $vq/2^\tau$. Since the marginal distribution of each $\mathbf{r}[i]$ is $q/2^\tau$ close to the uniform distribution, the actual probability that there's a query (x,ℓ) to RO_3 such that $x \in \{hk \,\|\, 1 \,\|\, \mathbf{r}[i] \mid 1 \le i \notin I\}$ is at most $2vq/2^\tau$. Hence the chance that Bad happens is at most

$$\frac{v(k)(q(k)+v(k))}{2^k} + q(k)v(k) \cdot \mathbf{Adv}^{\mathrm{rp}}_{\mathsf{NG},D}(k) + \frac{0.5v^2(k) + 2q(k)v(k)}{2^{\tau(k)}} .$$

Now, suppose that Bad doesn't happen. Note that in game G_{10}, the set I is independent of $\mathbf{m}$, and thus z_0 is identically distributed as $z_1 = \delta(\mathbf{m}, param)$, conditioning on $param, \mathbf{a}[I], \mathbf{m}[I], \boldsymbol{N}[I], \boldsymbol{xk}[U[I]]$. The adversary is given $\mathbf{r}[i]$, for $i \in I$, and it knows that $\mathcal{M}$ never queries $(1 \,\|\, \mathbf{r}[i], \ell)$. Thus $\Pr[G_{10}(\cdot) \Rightarrow 1 \mid \overline{\mathsf{Bad}}]$ is bounded by $1/2 + qv/2^{\mathsf{LT.il}} \le 1/2 + qv/2^\tau$, and thus

$$\begin{aligned}\Pr[G_{10}(k) \Rightarrow 1] \;&\le\; \frac{1}{2} + \frac{v(k)(q(k)+v(k))}{2^k} + q(k)v(k) \cdot \mathbf{Adv}^{\mathrm{rp}}_{\mathsf{NG},D}(k) \\ &\quad + \frac{0.5v^2(k) + 3q(k)v(k)}{2^{\tau(k)}} .\end{aligned}$$

Summing up,

$$\begin{aligned}\mathbf{Adv}^{\text{n-so-cpa}}_{\mathsf{NE1}[H,\mathsf{LT}],\mathsf{NG},A,\mathcal{M},\delta}(k) \;&=\; 2\Pr[G_1(k) \Rightarrow 1] - 1 \\ &\le\; 2\mathbf{Adv}^{\mathrm{ltdf}}_{\mathsf{LT},B}(k) + 8q(k)v(k) \cdot \mathbf{Adv}^{\mathrm{rp}}_{\mathsf{NG},D}(k) \\ &\quad + \frac{7v(k)(q(k)+v(k))}{2^k} + \frac{12v(k)(q(k)+v(k))}{2^{\tau(k)}} .\end{aligned}$$

B.3 Proof of Theorem 4.2

Let G_1 be the same game as $\text{N-SO-CCA}^{A,\mathcal{M},\delta}_{\mathsf{NE2}[H,\mathsf{LT}],\mathsf{NG}}$, but for clarity, we'll let $\overline{\mathrm{RO}}$ denote the adversary's interface to the random oracle RO. Let G_2 be the game that is identical to G_1, but it uses the procedures $\overline{\mathrm{RO}}$ and Dec as shown in Figure 28. Procedure $\overline{\mathrm{RO}}$ records the substrings $w[1, \mathsf{LT.il}(k)]$ in its queries $(hk \,\|\, 10 \,\|\, w, k)$ in a set S. Now, in the new Dec, if we are given a ciphertext $c = (trap, y, z)$, we check if any $r \in S$ can be a pre-image of $trap$. If yes then we'll output the corresponding message m (provided that $\overline{\mathrm{RO}}(hk \,\|\, 10 \,\|\, r \,\|\, m, k)$ matches z). Otherwise, we'll return $\perp$.

```
Algorithm $B.\mathrm{g}(state, \mathbf{y}^*, \mathbf{w}^*, param)$
For $i = 1$ to $|\mathbf{y}^*|$ do $(m, xk, N) \leftarrow \mathbf{y}^*[i]$ ; $\mathbf{a}^*[i] \leftarrow (a, xk, N)$ ; $\mathbf{m}[i] \leftarrow m$
Return $A.\mathrm{g}(state, \mathbf{m}^*, \mathbf{a}^*, param)$
```

```
Algorithm $\overline{\mathcal{M}}(1^k, param)$
$(\mathbf{m}, \mathbf{a}) \leftarrow\$ \mathcal{M}(1^k, param)$
For $i = 1$ to $|\mathbf{a}|$ do
    $(a, xk, N) \leftarrow \mathbf{a}[i]$ ; $\mathbf{w}[i] \leftarrow a$
    $\mathbf{y}[i] \leftarrow (\mathbf{m}[i], xk, N)$
Return $(\mathbf{y}, \mathbf{w})$
```

```
Algorithm $\overline{\mathsf{Rsmp}}(1^k, \mathbf{y}^*, \mathbf{w}^*, I, param)$
For $i = 1$ to $|\mathbf{y}^*|$ do
    $(m, xk, N) \leftarrow \mathbf{y}^*[i]$ ; $\mathbf{m}^*[i] \leftarrow m$
    $\mathbf{a}^*[i] \leftarrow (\mathbf{w}^*[i], xk, N)$
$z \leftarrow\$ \mathsf{Rsmp}(1^k, \mathbf{m}^*, \mathbf{a}^*, I, param)$
Return $z$
```

Figure 29: **Adversary B, constructed sampler $\overline{\mathcal{M}}$, and its partial resampling algorithm $\overline{\mathsf{Rsmp}}$ in the proof of Theorem 5.2, and also in the proof of Theorem 5.4.** Adversary B is constructed from an adversary A, and B.pg, B.cor, and B.f coincide with A.pg, A.cor, and A.f, respectively.

We now bound the gap between $\Pr[G_1 \Rightarrow 1]$ and $\Pr[G_2 \Rightarrow 1]$. The adversary can distinguish the two games if it can make a decryption query $c = (trap, y, z)$ such that the decryption oracle in game G_2 returns $\perp$, but that of game G_1 returns a non-$\perp$ answer. Let $r = \mathsf{LT.Inv}(td, trap)$. Since the decryption oracle in game G_2 returns $\perp$ on this query, the adversary must never query $\overline{\mathrm{RO}}(hk \,\|\, 10 \,\|\, r \,\|\, s, k)$ before, for any $s \in \{0,1\}^*$. First consider the case that the query is made before the adversary is given the ciphertexts $\mathbf{c}$. Since the decryption oracle in game G_1 returns a non-$\perp$ answer m for this query, we must have $z = \mathrm{RO}(hk \,\|\, 10 \,\|\, r \,\|\, m, k)$, which happens with probability at most 2^{-k}. Next consider the case that the query is made after the adversary is given the ciphertexts $\mathbf{c}$. Since the decryption oracle in game G_1 returns a non-$\perp$ answer m for this query, we must have $z = \mathrm{RO}(hk \,\|\, 10 \,\|\, r \,\|\, m, k)$. If there is no $i \in \{1, \ldots, v(k)\}$ such that $(r, m) = (\mathbf{r}[i], \mathbf{m}_1[i])$ then $z = \mathrm{RO}(hk \,\|\, 10 \,\|\, r \,\|\, m, k)$ with probability 2^{-k}, otherwise c is the same as $\mathbf{c}[i]$, which is forbidden. Hence

$$\Pr[G_1(k) \Rightarrow 1] - \Pr[G_2(k) \Rightarrow 1] \le \frac{p(k)}{2^k} \quad .$$

Now in game G_2, the adversary actually can compute the answers of DEC by itself. Each decryption query will cost two additional $\overline{\mathrm{RO}}$ queries. Hence we can assume that A makes no decryption queries, and instead makes at most $q + 2p$ random-oracle queries.

What's left is to prove the N-SO-CPA security of $\mathsf{NE2}[H, \mathsf{LT}]$ against an adversary A that makes at most $q + 2p$ queries. Let $Q = 2p + q + v$. Following the proof of Theorem 4.1, there are adversaries B and D of the claimed running time such that

$$\mathbf{Adv}^{\text{n-so-cpa}}_{\mathsf{NE2}[H,\mathsf{LT}],\mathsf{NG},A,\mathcal{M},\delta}(k) \le 2\mathbf{Adv}^{\text{ltdf}}_{\mathsf{LT},B}(k) + 8v(k)Q(k) \cdot \mathbf{Adv}^{\text{rp}}_{\mathsf{NG},D}(k) + \frac{7v(k)Q(k)}{2^k} + \frac{12v(k)Q(k)}{2^{\tau(k)}} \quad .$$

This completes the proof.

B.4 Proof of Theorem 5.2

Consider the following message sampler $\overline{\mathcal{M}}(1^k, param)$. It runs $\mathcal{M}(1^k, param)$ to generate $(\mathbf{m}, \mathbf{a})$. Since $\mathcal{M}$ is unpredictable, each $\mathbf{a}[i]$ can be parsed as (a_i, xk_i, N_i). Now the "messages" of $\overline{\mathcal{M}}$ is the vector $\mathbf{y}$, where each $\mathbf{y}[i] = (\mathbf{m}[i], xk_i, N_i)$, and the corresponding auxiliary information is a_i. The code of $\overline{\mathcal{M}}$ is given in Figure 29. Since $\mathcal{M}$ is (μ, d)-unpredictable, $\overline{\mathcal{M}}$ is (μ, d)-entropic. Let δ be a function such that $\mathcal{M}$ is δ-partially resamplable. Let $\overline{\delta}(\mathbf{y}, param)$ be the following function. It parses each $\mathbf{y}[i]$ as $(\mathbf{m}[i], xk_i, N_i)$, and then returns $\delta(\mathbf{m}, param)$. Then $\overline{\mathcal{M}}$ is $\overline{\delta}$-partially resamplable: given any δ-partial resampling algorithm Rsmp for $\mathcal{M}$, we can construct a $\overline{\delta}$-partial resampling algorithm $\overline{\mathsf{Rsmp}}$ for $\overline{\mathcal{M}}$ as in Figure 29. Let B be the adversary attacking the D-PKE scheme $\mathsf{DE1}$ in Section 3.2 as specified in Figure 29. Note that game $\text{HN-CPA-REAL}^{A,\mathcal{M},\delta}_{\mathsf{NE1}[H,\mathsf{LT}]}$ coincides with game $\text{D-CPA2-REAL}^{B,\overline{\mathcal{M}},\overline{\delta}}_{\mathsf{DE1}[H,\mathsf{LT}]}$, and game $\text{HN-CPA-IDEAL}^{A,\mathcal{M},\delta}_{\mathsf{NE1}[H,\mathsf{LT}]}$ coincides with $\text{D-CPA2-IDEAL}^{B,\overline{\mathcal{M}},\overline{\delta}}_{\mathsf{DE1}[H,\mathsf{LT}]}$. Hence

$$\mathbf{Adv}^{\text{hn-so-cpa}}_{\mathsf{NE1}[H,\mathsf{LT}],A,\mathcal{M},\delta}(\cdot) \le \mathbf{Adv}^{\text{d-so-cpa}}_{\mathsf{DE1}[H,\mathsf{LT}],B,\overline{\mathcal{M}},\overline{\delta}}(\cdot) \quad .$$

The claimed result then follows from Theorem 3.2.

```
$\textbf{Algorithm } B^{\text{Dec}}(1^k, pk)$
$(pk_{\mathrm{d}}, sk_{\mathrm{d}}) \leftarrow\$ \mathsf{DE.Kg}(1^k)$
$(J, t) \leftarrow\$ A^{\text{DecSim}}(1^k, pk)$
$state \leftarrow (pk_{\mathrm{d}}, sk_{\mathrm{d}}, t)$
Return $(J, state)$

$\textbf{Algorithm } B^{\text{Dec}}(state, \mathbf{c})$
$(pk_{\mathrm{d}}, sk_{\mathrm{d}}, t) \leftarrow state$
For $i = 1$ to $|\mathbf{c}|$ do $\overline{\mathbf{c}} \leftarrow \mathsf{DE.Enc}(pk_{\mathrm{d}}, \mathbf{c}[i])$
$(I, t) \leftarrow\$ A^{\text{DecSim}}(state, \overline{\mathbf{c}})$
$state \leftarrow (pk_{\mathrm{d}}, sk_{\mathrm{d}}, t)$ ; Return $(I, state)$

$\textbf{Algorithm } B^{\text{Dec}}(state, \boldsymbol{xk}^*)$
$(pk_{\mathrm{d}}, sk_{\mathrm{d}}, t) \leftarrow state$
$(param, \boldsymbol{N}, U, \boldsymbol{\sigma}, t) \leftarrow\$ A^{\text{DecSim}}(t, \boldsymbol{xk}^*)$
$state \leftarrow (pk_{\mathrm{d}}, sk_{\mathrm{d}}, t)$
Return $(param, \boldsymbol{N}, U, \boldsymbol{\sigma}, state)$

$\textbf{Algorithm } B^{\text{Dec}}(state, \mathbf{m}^*, \mathbf{a}^*, \boldsymbol{N}^*, \boldsymbol{xk}^*)$
$(pk_{\mathrm{d}}, sk_{\mathrm{d}}, t) \leftarrow state$
Return $A^{\text{DecSim}}(t, \mathbf{m}^*, \mathbf{a}^*, \boldsymbol{N}^*, \boldsymbol{xk}^*)$
```

```
$\textbf{Procedure } \text{DecSim}(c)$
$y \leftarrow \mathsf{DE.Dec}(sk_{\mathrm{d}}, c)$ ; $m \leftarrow \text{Dec}(y)$ ; Return $m$
```

Figure 30: **N-SO-CCA adversary B in the proof of Theorem 5.3**.

```
$\textbf{Algorithm } B.\mathrm{pg}(1^k)$
$param \leftarrow\$ A.\mathrm{pg}(1^k)$
$(pk_{\mathrm{n}}, sk_{\mathrm{n}}) \leftarrow\$ \mathsf{NE.Kg}(1^k)$
$pars \leftarrow (pk_{\mathrm{n}}, sk_{\mathrm{n}}, param)$
Return $pars$

$\textbf{Algorithm } B.\mathrm{cor}^{\text{Dec}}(pk_{\mathrm{d}}, \mathbf{c}, pars)$
$(pk_{\mathrm{n}}, sk_{\mathrm{n}}, param) \leftarrow pars$
$pk \leftarrow (pk_{\mathrm{d}}, pk_{\mathrm{n}})$
$(state, I) \leftarrow\$ A.\mathrm{cor}^{\text{DecSim}}(pk, \mathbf{c}, param)$
$t \leftarrow (state, pars)$ ; Return $(t, I)$

$\textbf{Algorithm } B.\mathrm{g}^{\text{Dec}}(t, \mathbf{y}^*, \mathbf{a}^*)$
$(state, pars) \leftarrow t$
$(pk_{\mathrm{n}}, sk_{\mathrm{n}}, param) \leftarrow pars$
For $i \leftarrow 1$ to $|\mathbf{y}^*|$ do
    $\mathbf{m}[i] \leftarrow \mathsf{NE.Dec}(sk_{\mathrm{n}}, \mathbf{y}[i])$
$\omega \leftarrow\$ A.\mathrm{g}^{\text{DecSim}}(state, \mathbf{m}^*, \mathbf{a}^*)$
Return $\omega$

$\textbf{Algorithm } B.\mathrm{f}(z, pars)$
$(pk_{\mathrm{n}}, sk_{\mathrm{n}}, param) \leftarrow pars$
$t \leftarrow\$ A.\mathrm{f}(z, param)$ ; Return $t$
```

```
$\textbf{Procedure } \text{DecSim}(c)$
$y \leftarrow \text{Dec}(c)$ ; $m \leftarrow \mathsf{NE.Dec}(sk_{\mathrm{n}}, y)$ ; Return $m$
```

Figure 31: **D-SO-CCA adversary B in the proof of Theorem 5.3.**

B.5 Proof of Theorem 5.3

For the first part, consider an arbitrary adversary A. Consider the adversary B in Figure 30 attacking NE. Note that due to the unique-ciphertext property of DE, adversary A can't force B to query its ciphertexts to its decryption oracle. Then game $\text{N-SO-CCA}^{B,\mathcal{M},\delta}_{\mathsf{NE},\mathsf{NG}}$ coincides with game $\text{N-SO-CCA}^{A,\mathcal{M},\delta}_{\mathsf{NE},\mathsf{NG}}$, and thus $\mathbf{Adv}^{\text{n-so-cca}}_{\mathsf{NE},\mathsf{NG},A,\mathcal{M},\delta}(\cdot) \le \mathbf{Adv}^{\text{n-so-cca}}_{\mathsf{NE},\mathsf{NG},B,\mathcal{M},\delta}(\cdot)$.

For the second part, consider an arbitrary adversary A and a message sampler $\mathcal{M}$. Consider the following message sampler $\overline{\mathcal{M}}(1^k, pars)$. It parses $param$ as a triple $(pk_{\mathrm{n}}, sk_{\mathrm{n}}, param)$, where pk_{n} and sk_{n} are public and secret keys for NE. It then runs $\mathcal{M}(1^k, param)$ to generate $(\mathbf{m}, \mathbf{a})$. Since $\mathcal{M}$ is unpredictable, each $\mathbf{a}[i]$ can be parsed as (a_i, xk_i, N_i). Now the "messages" of $\mathcal{M}$ is the vector $\mathbf{y}$, where each $\mathbf{y}[i] = \mathsf{NE.Enc}(pk_{\mathrm{n}}, xk_i, N_i, \mathbf{m}[i])$, and the corresponding auxiliary information is still $\mathbf{a}[i]$. The code of $\overline{\mathcal{M}}$ is given in Figure 13, which is in the proof of Theorem 5.1. Since $\mathcal{M}$ is (μ, d)-unpredictable, $\overline{\mathcal{M}}$ is $(\mathsf{NE.Guess}(\mu), d)$-entropic. Let δ be a function such that $\mathcal{M}$ is δ-partially resamplable. Let $\overline{\delta}(\mathbf{y}, pars)$ be the following function. It parses $pars$ as $(pk_{\mathrm{n}}, sk_{\mathrm{n}}, param)$, decrypts $\mathbf{m}[i] \leftarrow \mathsf{NE.Dec}(sk_{\mathrm{n}}, \mathbf{y}[i])$, and then returns $\delta(\mathbf{m}, param)$. Then $\overline{\mathcal{M}}$ is $\overline{\delta}$-partially resamplable: given any δ-partial resampling algorithm Rsmp for $\mathcal{M}$, we can construct a $\overline{\delta}$-partial resampling algorithm $\overline{\mathsf{Rsmp}}$ for $\overline{\mathcal{M}}$ as in Figure 13.

Now, consider the adversary B attacking DE as given in Figure 31. It targets message sampler $\overline{\mathcal{M}}$, with respect to function $\overline{\delta}$. Initially, $B.\mathrm{pg}(1^k)$ runs $param \leftarrow A(1^k)$, and then generates public and secret keys pk_{n} and sk_{n} for NE. It then outputs $pars \leftarrow (pk_{\mathrm{n}}, sk_{\mathrm{n}}, param)$. When $B.\mathrm{g}$ receives its "messages" $\mathbf{y}^*$, it extracts the secret key sk_{n} from its state and decrypts $\mathbf{m}^*[i] \leftarrow \mathsf{NE.Dec}(sk_{\mathrm{n}}, \mathbf{y}^*[i])$, and then gives $\mathbf{m}^*$ to $A.\mathrm{g}$ together with the auxiliary information $\mathbf{a}^*$. When $A.\mathrm{g}$ asks a decryption query c, adversary $B.\mathrm{g}$ uses its decryption oracle to decrypt c to get a ciphertext y of NE. It then uses sk_{n} to decrypt y, and returns the result to $A.\mathrm{g}$. Then game $\text{HN-CCA-REAL}^{A,\mathcal{M},\delta}_{\overline{\mathsf{NE}}}$ coincides with game $\text{D-CCA-REAL}^{B,\overline{\mathcal{M}},\overline{\delta}}_{\mathsf{DE}}$, and game $\text{HN-CCA-IDEAL}^{A,\mathcal{M},\delta}_{\overline{\mathsf{NE}}}$ coincides with $\text{D-CCA-IDEAL}^{B,\overline{\mathcal{M}},\overline{\delta}}_{\mathsf{DE}}$.

```
$\underline{\mathsf{DE.Kg}(1^k)}$
$(ek, td) \leftarrow_{\$} \mathsf{LT.IKg}(1^k)$
$hk \leftarrow_{\$} \{0,1\}^k$
$pk \leftarrow (hk, ek)$
$sk \leftarrow (hk, ek, td)$
Return $(pk, sk)$

$\underline{\mathsf{DE.Enc}(pk, m)}$
$(hk, ek) \leftarrow pk$ ; $r \leftarrow H(hk \,\|\, 00 \,\|\, m, \mathsf{LT.il}(k))$
$trap \leftarrow \mathsf{LT.Eval}(ek, r)$
$y \leftarrow H(hk \,\|\, 01 \,\|\, r, |m|) \oplus m$
$z \leftarrow H(hk \,\|\, 10 \,\|\, r \,\|\, m, k)$
Return $(trap, y, z)$
```

Figure 32: **D-PKE scheme $\mathsf{DE} = \mathsf{DE3}[H, \mathsf{LT}]$ in the proof of Theorem 5.4.** The decryption of DE is omitted, because it doesn't matter for D-SO-CPA security.

Hence $\mathbf{Adv}^{\text{hn-so-cca}}_{\mathsf{NE},A,\mathcal{M},\delta}(\cdot) \le \mathbf{Adv}^{\text{d-so-cca}}_{\mathsf{DE},B,\overline{\mathcal{M}},\overline{\delta}}(\cdot)$.

B.6 Proof of Theorem 5.4

Let G_1 be the same game as $\text{HN-SO-CCA}^{A,\mathcal{M},\delta}_{\mathsf{NE2}[H,\mathsf{LT}]}$, but for clarity, we'll let RO_1, RO_2, and RO_3 denote the interfaces to the random oracle of $(A.\text{pg}, \mathcal{M})$, $(A.\text{cor}, A.\text{g})$, and $A.\text{f}$ respectively. Since $A.\text{cor}$ and $A.\text{g}$ share a state, wlog, assume that the joint adversary $(A.\text{cor}, A.\text{g})$ doesn't repeat a prior random-oracle query. Moreover, in the interface RO_2, for bookkeeping purpose, we record the queries of the form $(hk \,\|\, 10 \,\|\, x, k)$ and store $x[1, \mathsf{LT.il}(k)]$ in a set S. Game G_2 is identical to game G_1, but using the following implementation of the decryption oracle $\textsc{Dec}$. On input $c = (trap, y, z)$, algorithm $\textsc{Dec}(c)$ checks if there is some $r \in S$ such that $\mathsf{LT.Eval}(ek, r) = trap$. If yes then it outputs

$$m \leftarrow \text{RO}(hk \,\|\, 01 \,\|\, r, |y|) \oplus y,$$

provided that $z = \text{RO}(hk \,\|\, 10 \,\|\, r \,\|\, m, k)$. Otherwise, it outputs $\bot$. We now bound the gap between $\Pr[G_1 \Rightarrow 1]$ and $\Pr[G_2 \Rightarrow 1]$. Assume that $A.\text{pg}$ and $\mathcal{M}$ never query $\text{RO}_1(hk \,\|\, \cdot, \cdot)$. This happens with probability at least $1 - q(k)/2^k$. The adversary A can distinguish the two games if it can make a decryption query $c = (trap, y, z)$ such that the decryption oracle in game G_2 returns $\bot$, but that of game G_1 returns a non-$\bot$ answer. Let $r = \mathsf{LT.Inv}(td, trap)$. Since the decryption oracle in game G_2 returns $\bot$ on this query, the adversary $(A.\text{cor}, A.\text{g})$ must never query $\text{RO}_2(hk \,\|\, 10 \,\|\, r \,\|\, s, k)$ before, for any $s \in \{0,1\}^*$. Since the decryption oracle in game G_1 returns a non-$\bot$ answer m for this query, we must have $z = \text{RO}(hk \,\|\, 10 \,\|\, r \,\|\, m, k)$. If there is no $i \in \{1, \ldots, v(k)\}$ such that $(r, m) = (\mathbf{r}[i], \mathbf{m}_1[i])$ then $z = \text{RO}(hk \,\|\, 10 \,\|\, r \,\|\, m, k)$ with probability 2^{-k}, otherwise c is the same as $\mathbf{c}[i]$, which is forbidden. Hence

$$\Pr[G_1(k) \Rightarrow 1] - \Pr[G_2(k) \Rightarrow 1] \le \frac{p(k)}{2^k} \enspace .$$

Now in game G_2, the adversary actually can compute the answers of $\textsc{Dec}$ by itself. Each decryption query will cost two additional RO queries. Hence we can assume that A makes no decryption queries, and instead makes at most $q + 2p$ random-oracle queries. So the next step is to bound the HN-SO-CPA security of $\mathsf{NE2}$, against an adversary A that makes at most $Q = q + 2p$ random-oracle queries. Consider the following message sampler $\overline{\mathcal{M}}(1^k, param)$. It runs $\mathcal{M}(1^k, param)$ to generate $(\mathbf{m}, \mathbf{a})$. Since $\mathcal{M}$ is unpredictable, each $\mathbf{a}[i]$ can be parsed as (a_i, xk_i, N_i). Now the "messages" of $\mathcal{M}$ is the vector $\mathbf{y}$, where each $\mathbf{y}[i] = (\mathbf{m}[i], xk_i, N_i)$, and the corresponding auxiliary information is a_i. The code of $\overline{\mathcal{M}}$ is given in Figure 29. Since $\mathcal{M}$ is (μ, d)-unpredictable, $\overline{\mathcal{M}}$ is (μ, d)-entropic. Let δ be a function such that $\mathcal{M}$ is δ-partially resamplable. Let $\overline{\delta}(\mathbf{y}, param)$ be the following function. It parses each $\mathbf{y}[i]$ as $(\mathbf{m}[i], xk_i, N_i)$, and then returns $\delta(\mathbf{m}, param)$. Then $\overline{\mathcal{M}}$ is $\overline{\delta}$-partially resamplable: given any δ-partial resampling algorithm Rsmp for $\mathcal{M}$, we can construct a $\overline{\delta}$-partial resampling algorithm $\overline{\mathsf{Rsmp}}$ for $\overline{\mathcal{M}}$ as in Figure 29. Let B be the adversary attacking the D-PKE scheme $\mathsf{DE3}$ as specified in Figure 29. Note that game $\text{HN-CPA-REAL}^{A,\mathcal{M},\delta}_{\mathsf{NE2}[H,\mathsf{LT}]}$ coincides with game $\text{D-CPA2-REAL}^{B,\overline{\mathcal{M}},\overline{\delta}}_{\mathsf{DE3}[H,\mathsf{LT}]}$, and game $\text{HN-CPA-IDEAL}^{A,\mathcal{M},\delta}_{\mathsf{NE2}[H,\mathsf{LT}]}$ coincides with $\text{D-CPA2-IDEAL}^{B,\overline{\mathcal{M}},\overline{\delta}}_{\mathsf{DE3}[H,\mathsf{LT}]}$. Then

$$\mathbf{Adv}^{\text{hn-so-cpa}}_{\mathsf{NE2}[H,\mathsf{LT}],A,\mathcal{M},\delta}(\cdot) \le \mathbf{Adv}^{\text{d-so-cpa2}}_{\mathsf{DE3}[H,\mathsf{LT}],B,\overline{\mathcal{M}},\overline{\delta}}(\cdot) \enspace .$$

Following the proof of Theorem 3.2, we can construct an adversary D of the claimed running time such that

$$\mathbf{Adv}^{\text{d-so-cpa2}}_{\mathsf{DE3}[H,\mathsf{LT}],B,\overline{\mathcal{M}},\overline{\delta}}(k) \le \frac{4Q(k)}{2^k} + \frac{4Q(k)v(k)}{2^{\mu(k)}} + \frac{v(k)(v(k) + 4Q(k))}{2^{\tau(k)}} + 2\mathbf{Adv}^{\text{ltdf}}_{\mathsf{LT},D}(k).$$

Summing up yields the claimed result.