What about Bob? The Inadequacy of CPA Security for Proxy Reencryption

Aloni Cohen*

MIT
aloni@mit.edu

Abstract. In the simplest setting of proxy reencryption, there are three parties: Alice, Bob, and Polly (the proxy). Alice keeps some encrypted data that she can decrypt with a secret key known only to her. She wants to communicate the data to Bob, but not to Polly (nor anybody else). Using proxy reencryption, Alice can create a *reencryption key* that will enable Polly to reencrypt the data for Bob's use, but which will not help Polly learn anything about the data.

There are two well-studied notions of security for proxy reencryption schemes: security under chosen-plaintext attacks (CPA) and security under chosen-ciphertext attacks (CCA). Both definitions aim to formalize the security that Alice enjoys against both Polly and Bob.

In this work, we demonstrate that CPA security guarantees much less security against Bob than was previously understood. In particular, CPA security does not prevent Bob from learning Alice's secret key after receiving a single honestly reencrypted ciphertext. We also show that an existing construction of CPA secure proxy reencryption suffers from this type of weakness. As a result, CPA security provides scant guarantees in common applications.

We propose security under honest reencryption attacks (HRA), a strengthening of CPA security that better captures the goals of proxy reencryption. In applications, HRA security provides much more robust security. We identify a property of proxy reencryption schemes that suffices to amplify CPA security to HRA security and show that two existing proxy reencryption schemes are in fact HRA secure.

Keywords: proxy reencryption · definitions · public-key cryptography

* Supported by Facebook Fellowship 2018, NSF GRFP, NSF MACS CNS-1413920, DARPA IBM W911NF-15-C-0236, and Simons Investigator Award Agreement Dated 6-5-12. We would like to thank Rio LaVigne, Akshay Degwekar, Shafi Goldwasser, Vinod Vaikuntanathan, and anonymous reviewers for their helpful feedback.

1 Introduction

Consider three parties: Alice, Bob, and Polly Proxy. Alice keeps encrypted data (created with a public key) that she can decrypt with a secret key known only to her. She wants to communicate some of the data to Bob, but not to Polly (nor anybody else). Assuming Alice and Polly know Bob's public key, how can she communicate the data to him?

If she is willing to entrust Bob with all her secrets, past and future, Alice might try to tell Bob her secret decryption key by encrypting it using Bob's public key. We call this the *Trivial Scheme*. If she does not have such trust in Bob, Alice can instead decrypt the data, and reencrypt it using Bob's public key. But what if Alice does not want to do the work of decrypting and reencrypting large amounts of data?

Proxy reencryption (PRE) provides an elegant solution: Alice creates and gives to Polly a *reencryption key* that will enable Polly to reencrypt her data under Bob's public key for his use, but that will not reveal the data to Polly. Proxy reencryption guarantees that Bob can recover the data uncorrupted (correctness) and that Polly cannot learn anything about Alice's data (security). The most widely-studied model of security for proxy reencryption is called *CPA security*, named after the corresponding notion from standard encryption on which it is based.

But what about Bob? As already observed, if we do not require any security against Bob, proxy reencryption is trivial: Alice uses the Trivial Scheme, simply sending Bob her encrypted secret key. This is undesirable, unsatisfying, and insufficient for a number of supposed applications of proxy reencryption (Section 2).

Surprisingly, *the Trivial Scheme is a CPA secure proxy reencryption scheme* when the public key encryption scheme used is circularly secure [BHHO08]! Bob completely learns Alice's secret key, and circular security is used only to prove security against a malicious Polly. Furthermore, the CPA-security of any proxy reencryption scheme remains uncompromised if Polly attaches the reencryption key to every reencrypted ciphertext sent to Bob, even though this would enable Bob to decrypt messages encrypted under Alice's public key (Section 3.1).

These "constructions" of CPA-secure proxy reencryption—original to this work—demonstrate the inadequacy of CPA security for proxy reencryption. If they had been observed previously, perhaps CPA security would not have gained the traction that it has.

Throughout this work, we use CPA (respectively, CCA and HRA) to refer to the security notion for proxy reencryption, and Enc-CPA (resp., Enc-CCA) to refer to the security notion for standard encryption. We restrict our attention to *unidirectional* proxy reencryption, where the reencryption key allows Alice's ciphertexts to be reencrypted to Bob's key, but not the reverse. In a bidirectional scheme, Bob—using his own secret key and Alice's public key— is able compute the Alice-to-Bob reencryption key himself; thus a lack of security against Bob is inherent.

1.1 CPA and CCA Security of Proxy Reencryption

First considered by Blaze, Bleumer, and Strauss [BBS98], proxy reencryption has received significant and continuous attention in the last decade, including definitions [ID03, AFGH06, CH07, NAL15], number-theoretical constructions [ABH09, LV08, CWYD10], lattice-based constructions [Gen09, ABW+13, PWA+16, FL17], implementations [LPK10, HHY11, PRSV17, BPR+17], and early success in program obfuscation [HRSV07, CCL+14].

Adapting notions from standard encryption, this literature considers two main indistinguishability-based security notions for proxy reencryption: security under *chosen plaintext attacks (CPA)* [ABH09] and *chosen ciphertext attacks (CCA)* [CH07]. While CCA security is considered the gold-standard, CPA security has received significant attention [AFGH06, ABH09, HRSV07], especially in latticed-based constructions [Gen09, ABW+13, PWA+16, PRSV17]. CPA security has been used as a testing ground for new techniques for proxy reencryption and in settings where efficiency concerns make the added security of CCA undesirable.

We now briefly describe the definitions of CPA and CCA security for proxy reencryption, with the goal of communicating the underlying intuition. For this informal description, we restrict our attention to the limited three party setting of Alice, Bob, and Polly and strip away many of the complexities of the full definition. For a full definitions of CPA and CCA security, see Definitions 3 and 10 respectively.

Both notions are typically defined using a security game between an adversary and a challenger in which the adversary's task is to distinguish between encryptions of two messages. Both notions allow the adversary to corrupt either Bob (learning $\mathsf{sk}_{\mathsf{bob}}$) or Polly (learning the reencryption key rk). CCA and CPA security differ in the additional power granted to the adversary.

CCA security grants the adversary access to two oracles:

- $\mathcal{O}_{\mathsf{Dec}}$: The decryption oracle takes as input a ciphertext along with the public key of either Alice or Bob, and outputs the decryption of the ciphertext using the corresponding secret key.
- $\mathcal{O}_{\mathsf{ReEnc}}$: The reencryption oracle takes as input a ciphertext $\mathsf{ct}_{\mathsf{alice}}$ and outputs a reencrypted ciphertext $\mathsf{ct}_{\mathsf{bob}}$.

These oracles come with restrictions to prevent the adversary from simply reencrypting or decrypting the challenge ciphertext, adapting replayable chosen-ciphertext security of standard encryption (Enc-CCA) in the natural way.

CPA security of proxy reencryption, however, removes both oracles.[1] Why? First, to adapt chosen-plaintext security from standard encryption (Enc-CPA) to proxy reencryption, we must of course do away with $\mathcal{O}_{\mathsf{Dec}}$. Secondly, it seems

[1] This description is an oversimplification. In the many party setting, the adversary has access to a reencryption oracle which will reencrypt ciphertexts between two uncorrupted parties or between two corrupted parties, but not from an honest party to a corrupted party.

we must also remove $\mathcal{O}_{\mathsf{ReEnc}}$: otherwise, by corrupting Bob it seems that the adversary can use the combination of $\mathcal{O}_{\mathsf{ReEnc}}$ and $\mathsf{sk}_{\mathsf{bob}}$ to simulate $\mathcal{O}_{\mathsf{Dec}}$. Removing both decryption and reencryption oracles, according to [ABH09], naturally extends the Enc-CPA security to proxy reencryption, yielding CPA security.

As we observe in this work, a natural definition is not always a good definition. Not only is the above intuition for removing $\mathcal{O}_{\mathsf{ReEnc}}$ false (Theorem 6), but CPA security as defined above guarantees little against a honest-but-curious Bob, even under *normal operation*. The definition only requires that the adversary will not win the game as long as it never sees any reencrypted ciphertexts. It guarantees nothing if Bob sees even a single reencrypted ciphertext. This vulnerability is not purely theoretical: in the CPA secure scheme of [PRSV17], Bob can recover Alice's secret key with significant probability from a single reencrypted ciphertext (Theorem 4).

This makes CPA security ill-suited for the most commonly cited applications of proxy reencryption, including forwarding of encrypted email and single-writer, many-reader encrypted storage (Section 2). CPA security is inadequate for proxy reencryption and must be replaced.

1.2 Security Against Honest Reencryption Attacks

What minimal guarantees should proxy reencryption provide? First, it should offer security against a dishonest proxy Polly when Alice and Bob are honest and using the proxy reencryption as intended. Polly's knowledge of a reencryption key from Alice to Bob (or vice versa) should not help her learn anything about the messages underlying ciphertexts encrypted under $\mathsf{pk}_{\mathsf{alice}}$ or $\mathsf{pk}_{\mathsf{bob}}$. Such security against the corrupted proxy is guaranteed by CPA.

Second, proxy reencryption should offer security against a dishonest receiver Bob when Alice and Polly are honest and using the proxy reencryption as intended. Bob's knowledge of *honestly reencrypted ciphertexts* (that were honestly generated to begin with) should not help him learn anything about the messages underlying other ciphertexts encrypted under $\mathsf{pk}_{\mathsf{alice}}$ that have not been reencrypted. As we show in this work, such security against the corrupted receiver is not guaranteed by CPA.

Generalizing these dual guarantees to many possibly colluding parties, we want security as long as the adversary only sees honestly reencrypted ciphertexts. In Section 4, we formalize this notion as proxy reencryption security against *honest reencryption attacks (HRA)*. Recall that CCA security provides the adversary with both $\mathcal{O}_{\mathsf{Dec}}$ and $\mathcal{O}_{\mathsf{ReEnc}}$ while CPA provides neither oracle. In contrast, HRA security provides the adversary with a restricted reencryption oracle which will only reencrypt honestly generated ciphertexts.

By guaranteeing security of both kinds described above, HRA is a strengthening of CPA security that better captures our intuitions for security of proxy reencryption. Furthermore, HRA guarantees more meaningful security in the most common applications of proxy reencryption (Section 4.1). HRA security is an appropriate goal when developing new techniques for proxy reencryption and in settings where full CCA security is undesirable or out of reach.

Security of Existing Schemes. Can we construct a proxy reencryption scheme that is HRA secure? HRA security is a strict strengthening of CPA security,so it is not immediately clear that any existing constructions are HRA secure without redoing the proofs from scratch. Indeed, the CPA secure scheme of [PRSV17] is not HRA secure (Theorem 4).

In Section 5, we identify a property—*reencryption simulatablity*—which is sufficient to boost CPA security to HRA security. Very roughly, reencryption simulatability means that reencrypted ciphertexts resulting from computing $\mathsf{ReEnc}(\mathsf{rk}_{\mathsf{alice}\to\mathsf{bob}}, \mathsf{ct}_{\mathsf{alice}})$ can be simulated without knowledge of the secret key $\mathsf{sk}_{\mathsf{alice}}$ (but with knowledge of the plaintext message $\mathbf{m}$). Reencryption simulatability allows a reduction with access to the CPA oracles to efficiently implement the honest reencryption oracle, thereby reducing HRA security to CPA security.

We the examine the simple construction of proxy reencryption from any fully-homomorphic encryption [Gen09], and the pairing-based construction of [AFGH06]. In the first case, if the fully-homomorphic encryption secure is circuit private, then the resulting proxy reencryption scheme is reencryption simulatable. In the second case, rerandomizing reencrypted ciphertexts suffices for reencryption simulation.[2]

1.3 Related Work

The below mentioned works are just the most directly relevant. There is by now an extensive research literature on proxy reencryption, presenting a zoo of definitions. There have been three main approaches to defining security: CPA, CCA, and (to a much lesser extent) obfuscation-based. The CPA notion, in one form or another, is by far the most well studied. In this work, we make the deliberate choice to address the core CPA definition, not to present an ultimate definition of security for proxy reencryption nor to address the vast array of different criticisms or strengthenings of CPA security that have been or may be considered. We hope that doing so will make these ideas more understandable and adaptable.

RIND-CPA Security In concurrent and independent work defining and constructing forward-secure proxy reencryption, Derler, Krenn, Lorünser, Ramacher, Slamanig, and Striecks identify the same problem with CPA security as discussed in this work [DKL+18, Definition 14]. As in our work, they address the problem with CPA security by defining a new security notion—RIND-CPA security—which expands the power of the adversary. They additionally separate RIND-CPA and CPA security with a construction that is essentially our Concatenation Scheme.

However, this is where the resemblance between [DKL+18] and our work ends. In the RIND-CPA game offered by [DKL+18], the adversary gets access

[2] While we don't examine every pairing-based construction of proxy reencryption, we suspect that rerandomizing reencryption will suffice for reencryption simulation in many, if not all.

to an reencryption oracle that works on all inputs (not just honestly generated ones), but only before the challenge ciphertext is generated.[3] In contrast, HRA allows reencryption both before and after the challenge, but only for honestly generated ciphertexts.

RIND-IND is inadequate as a replacement for CPA security in the research literature: its usefulness in applications is unclear, and it appears too strong to provide a useful testing ground for the development of new techniques for constructing proxy reencryption.

In the course of normal operation of a proxy reencryption in applications, an adversarial party will typically see many reencrypted ciphertexts. These ciphertexts may come at any time—both before and after other ciphertexts whose contents should remain secret. HRA is meaningful in many such applications—many more than CPA security. But because RIND-CPA restricts the reencryption oracle to the period before the challenge ciphertext, its usefulness in applications is not clear.

The challenge of proving CCA security for encryption (proxy or otherwise) is demonstrating that an adversary cannot use dishonestly generated, malformed ciphertexts to win in the security game. In this respect, RIND-CPA security is much more akin to CCA security than to CPA security. HRA, on the other hand, makes minimal assumptions about the distribution of plaintext messages by allowing the adversary to choose messages itself, just as in Enc-CPA for standard encryption.

Appendix C discusses RIND-CPA security in more depth, expanding on the arguments above and proving that RIND-CPA and HRA security are incomparable.

Subsequent Work Two subsequent works continue the study of HRA secure proxy reencryption. Fuchsbauer, Kamath, Klein, and Pietrzak study CPA and HRA secure proxy reencryption in an adaptive corruption model [FKKP18]. As in our work, they prove the HRA security of their construction by first proving CPA security and then lifting it to full HRA security using a version of reencryption simulatability.

More recently, Dottling and Nishimaki study the problem of converting ciphertexts between different public-key encryption schemes, a problem they call universal proxy reencryption [DN18]. They define security by extending HRA security to the universal setting. [DN18] extends Theorem 5 to show that a computational version of reencryption simulatability suffices to lift CPA to HRA security. However, they prove HRA security directly rather, finding that proving computational reencryption simulatability is not much more simple than proving HRA security itself.

Other Related Work Our dual-guarantee conception of proxy reencryption security mirrors the security requirements of what Ivan and Dodis call CPA

[3] Appendix B discusses the related definition of $\mathsf{IND\text{-}CCA}_{0,1}$ security from [NAL15]

security [ID03]. Their notion differs substantially from what is now referred to by that name. The [ID03] conception of CPA security is only defined in a proof in the appendix of that work and seems to have been completely overlooked by the later works proposing the modern notion of CPA security (beginning with [AFGH06] and then in its present form in [ABH09]). If, however, Ivan and Dodis had undertaken to revisit proxy reencryption after [ABH09], they might have proposed a security definition similar to HRA.

In [NAL15], Nuñez, Agudo, and Lopez provide a parameterized family of CCA-type attack models for proxy reencryption. Their weakest model corresponds to CPA security and their strongest to full CCA security. That work is partially a response to a claimed construction of CCA-1 secure proxy reencryption in a security model that does not allow reencryption queries. [NAL15] provide an attack on the construction in the presence of a reencryption oracle consisting of carefully constructed, dishonestly generated queries which leak the reencryption key. They do not consider restricting the reencryption oracle in the security game to honestly generated ciphertexts. We discuss [NAL15] further in Appendix B.3.

Finally, a parallel line of work initiated by Hohenberger, Rothblum, shelat, Vaikuntanathan which studies proxy reencryption using an obfuscation-based definition that is incomparable to CPA security [HRSV07]. Their definition requires that the functionality of the obfuscated reencryption circuit be statistically close to that of the ideal reencryption functionality: namely, that $\mathsf{ReEnc}(\mathsf{rk}_{i\to j}, \mathsf{Enc}(\mathsf{pk}_i, \mathbf{m})) \approx_s \mathsf{Enc}(\mathsf{pk}_j, \mathbf{m})$. Thus the definition of [HRSV07] (and even the relaxed correctness found in [CCL+14]) imply *reencryption simulatability* defined in Section 5.

1.4 Organization

We begin by discussing applications of proxy reencryption and identifying the weaknesses of CPA security in those applications (Section 2). Then we present the existing CPA security definition and further demonstrate its weaknesses with two new schemes: the Trivial Scheme and Concatentation Scheme (Section 3). We propose a new security notion to overcome those weaknesses: security against honest reenecryption attacks (HRA) (Section 4). We examine the relationship between CPA and HRA security and the HRA security (or insecurity) of existing reencryption schemes (Section 5). The appendix provides additional discussion of the Trivial Scheme (Appendix A), comparison between HRA and RIND-CPA security (Appendix C), and CCA security (Appendix B).

2 Insufficiency of CPA Security for Applications

In Section 3, we recall the definition of CPA security of proxy reencryption from [ABH09] and formalize the Trivial Scheme from the introduction satisfying the notion. In the Trivial Scheme, Bob learns Alice's secret key after receiving a single reencrypted ciphertext.

We are faced with a choice: accept the existing definition of CPA security, or reject it and seek a definition that better captures our intuitions. In support of the latter, we describe a number of applications of proxy reencryption proposed in the literature for which CPA security (as implemented by the Trivial Scheme) is potentially unsatisfactory, but for which full CCA security may not always be necessary.[4] We revisit these applications in Section 4.1 after proposing our new security notion.

Encrypted Email Forwarding [BBS98, Jak99, AFGH06]. A common suggestion, forwarding of encrypted email without requiring the sender's participation might be desirable for temporary delegation during a vacation [Jak99] or for spam filtering [AFGH06]. Does the Trivial Scheme suffice? The Trivial Scheme enables Bob, the receiver of Alice's forwarded (and reencrypted) email, to recover Alice's secret key. If Alice trusts Bob enough to use the Trivial Scheme, she could instead reveal her secret key. The Trivial Scheme might be preferable in very specific trust or interaction models, but it does not offer meaningful security against Bob if Alice only wishes to forward a subset of emails (for example, from particular senders or during a specific time period).

Key Escrow [ID03]. Similar to email forwarding, Ivan and Dodis describe the application of key escrow as follows: "The problem is to allow the law enforcement agency to read messages encrypted for a set of users, for a limited period of time, without knowing the users' secrets. The solution is to locate a key escrow agent between the users and the law enforcement agency, such that it controls which messages are read by the law enforcement agencies." As in email forwarding, the "for a limited period of time" requirement suggests that Ivan and Dodis would not have been satisfied with the Trivial Scheme.[5]

Single-Writer, Many-Reader Encrypted Storage [AFGH06, KHP06, LPK10, PRSV17]. Under different monikers (including DRM and publish/subscribe systems), these works describe systems in which a single privileged writer encrypts data and determines an access control policy for readers. A semi-honest proxy server is entrusted with reencryption keys and is

[4] We might also appeal for support to [ID03], the only paper in the proxy reencryption literature of which we are aware adopting a security definition providing a reencryption oracle without a decryption oracle. One could look to the originators of proxy reencryption for guidance, but the shortcoming we identify does not manifest in the original setting of [BBS98] (there is only Alice and Bob; there is no Proxy). It is therefore of little help.

[5] Note that Ivan and Dodis do not adopt the CPA definition used elsewhere, but a definition much closer to our own. There is no gap between their security guarantees and the requirements of their briefly-described application.

Though primarily focused on the setting where the key escrow agent enforces the limited time requirement by eventually refusing to reencrypt, [ID03] considers the possibility of dividing time into epochs and enforcing the time limitation technically. Such a proxy reencryption is called *temporary* in [AFGH06]. We do not discuss temporary proxy reencryption further.

tasked with enforcing the access control policy. Whether the Trivial Scheme suffices for these applications depends on what sort of access control policies are envisioned. If the access is *all or nothing* (i.e., all readers may access all data), the Trivial Scheme suffices; if the access is *fine grained* (i.e., each reader may access only a specific subset of the data), then it does not. Existing works are often unclear on which is envisioned.

For completeness, we mention that CPA security does suffice for two important applications of proxy reencryption: namely, key rotation for encrypted cloud storage [BLMR13, EPRS17] and fully homomorphic encryption [Gen09].

3 Security Against Chosen Plaintext Attacks

In this section, we recall the definition of CPA security for proxy reencryption and illustrate its shortcomings. We begin with the definitions of syntax, correctness, and CPA security from [ABH09, Definition 2.2] (with minor changes in notation and presentation, and the change noted in Remark 1 at the end of this subsection). The syntax and correctness requirements are common to CPA, HRA, and CCA security.

For the sake of concreteness, simplicity, and brevity, we restrict the discussion to unidirectional, single-hop schemes. In multi-hop schemes, reencryption keys $\mathsf{rk}_{A\to B}$ and $\mathsf{rk}_{B\to C}$ can be used to reencrypt a ciphertext ct_A from pk_A to pk_C. In single-hop schemes, they cannot. Single-hop or multi-hop schemes each have their benefits and drawbacks, and works typically focus on one or the other notion.[6] To the best of our knowledge, our observations and results can all be adapted to the multi-hop setting.

Definition 1 (Proxy Reencryption: Syntax [ABH09]). *A proxy reencryption scheme for a message space $\mathcal{M}$ is a tuple of algorithms* $\mathsf{PRE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{ReKeyGen}, \mathsf{Enc}, \mathsf{ReEnc}, \mathsf{Dec})$*:*

[6] The literature is divided about whether "single-hop" is merely a correctness property (i.e., able to reencrypt at least once, but agnostic about whether reencrypting more than once is possible) or if it is also a security property (i.e., a ciphertext can be reencrypted once, but never twice). This distinction manifests in the security definition. In works that consider only single-hop correctness [AFGH06, ABH09, HRSV07, NAL15], the oracle $\mathcal{O}_{\mathsf{ReKeyGen}}$ in the security game will not accept queries from honest users to corrupted users (i.e., (i, j) such that $i \in \mathsf{Hon}$ and $j \in \mathsf{Cor}$). We adopt this formalism in Definitions 3 and 5 for simplicity of presentation only.

In works that consider single-hop security [LV08, CWYD10, FL17], the oracle will answer such queries, but the challenge ciphertext must be encrypted under the key of an honest user i^* for which no such reencryption key was generated (which can be formalized in a number of ways).

This work adopts the simplest model, requiring only one hope of correctness, but neither requiring nor forbidding additional functionality.

$\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$. *On input security parameter* 1^λ*, the* setup *algorithm outputs the public parameters* pp.

$\mathsf{KeyGen}(\mathsf{pp}) \to (\mathsf{pk}, \mathsf{sk})$. *On input public parameters, the* key generation *algorithm outputs a public key* pk *and a secret key* sk. *For ease of notation, we assume that both* pk *and* sk *include* pp *and refrain from including* pp *as input to other algorithms.*

$\mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j) \to \mathsf{rk}_{i\to j}$. *On input a secret key* sk_i *and a public key* pk_j*, where* $i \neq j$*, the* reencryption key generation *algorithm outputs a reencryption key* $\mathsf{rk}_{i\to j}$.

$\mathsf{Enc}(\mathsf{pk}_i, \mathbf{m}) \to \mathsf{ct}_i$. *On input a public key* pk_i *and a message* $\mathbf{m} \in \mathcal{M}$*, the* encryption algorithm *outputs a ciphertext* ct_i.

$\mathsf{ReEnc}(\mathsf{rk}_{i\to j}, \mathsf{ct}_i) \to \mathsf{ct}_j$. *On input a reencryption key from* i *to* j $\mathsf{rk}_{i\to j}$ *and a ciphertext* ct_i*, the* reencryption algorithm *ouputs a ciphertext* ct_j *or the error symbol* $\perp$.

$\mathsf{Dec}(\mathsf{sk}_j, \mathsf{ct}_j) \to \mathbf{m}$. *Given a secret key* sk_j *and a ciphertext* ct_j*, the* decryption *algorithm outputs a message* $\mathbf{m} \in \mathcal{M}$ *or the error symbol* $\perp$.

Definition 2 (Proxy Reencryption: Correctness [ABH09]). *A proxy reencryption scheme* PRE *is correct with respect to message space* $\mathcal{M}$ *if for all* $\lambda \in \mathbb{N}$*,* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*, and* $\mathbf{m} \in \mathcal{M}$*:*

1. *for all* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$*:*

$$\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, \mathbf{m})) = \mathbf{m}.$$

2. *for all* $(\mathsf{pk}_i, \mathsf{sk}_i), (\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$*,* $\mathsf{rk}_{i\to j} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$*:*

$$\mathsf{Dec}(\mathsf{sk}_j, \mathsf{ReEnc}(\mathsf{rk}_{i\to j}, \mathsf{Enc}(\mathsf{pk}_i, \mathbf{m}))) = \mathbf{m}.$$

Security is modeled by a game played by an adversary $\mathcal{A}$. $\mathcal{A}$ has the power to corrupt a set of users Cor (learning their secret keys) while learning only the public keys for a set of honest users Hon. Additionally, $\mathcal{A}$ may reencrypt ciphertexts (either by getting a reencryption key or calling a reencryption oracle) between pairs of users in Hon or in Cor, or from Cor to Hon, but not from Hon to Cor. This is in contrast to the simplified three-party setting discussed in the introduction, where the $\mathcal{A}$ could not reencrypt whatsoever.

Definition 3 (Proxy Reencryption: Security Game for Chosen Plaintext Attacks (CPA) [ABH09]). *Let* λ *be the security parameter and* $\mathcal{A}$ *be an oracle Turing machine. The CPA game consists of an execution of* $\mathcal{A}$ *with the following oracles. The game consists of three phases, which are executed in order. Within each phase, each oracle can be executed in any order,* $\mathrm{poly}(\lambda)$ *times, unless otherwise specified.*

Phase 1:

Setup: *The public parameters are generated and given to* $\mathcal{A}$*. A counter* $\mathsf{numKeys}$ *is initialized to 0, and the sets* Hon *(of honest, uncorrupted indices) and* Cor *(of corrupted indices) are initialized to be empty. This oracle is executed first and only once.*

Uncorrupted Key Generation: *Obtain a new key pair* $(\mathsf{pk}_{\mathsf{numKeys}}, \mathsf{sk}_{\mathsf{numKeys}}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ *and give* $\mathsf{pk}_{\mathsf{numKeys}}$ *to* $\mathcal{A}$*. The current value of* numKeys *is added to* Hon *and* numKeys *is incremented.*

Corrupted Key Generation: *Obtain a new key pair* $(pk_{\mathsf{numKeys}}, \mathsf{sk}_{\mathsf{numKeys}}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ *and given to* $\mathcal{A}$*. The current value of* numKeys *is added to* Cor *and* numKeys *is incremented.*

Phase 2: For each pair $i, j \leq \mathsf{numKeys}$*, compute the reencryption key* $\mathsf{rk}_{i\to j} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$.

Reencryption Key Generation $\mathcal{O}_{\mathsf{ReKeyGen}}$: *On input* (i, j) *where* $i, j \leq \mathsf{numKeys}$*, if* $i = j$ *or if* $i \in \mathsf{Hon}$ *and* $j \in \mathsf{Cor}$*, output* $\perp$*. Otherwise return the reencryption key* $\mathsf{rk}_{i\to j}$.

Reencryption $\mathcal{O}_{\mathsf{ReEnc}}$: *On input* (i, j, ct_i) *where* $i, j \leq \mathsf{numKeys}$*, if* $i = j$ *or if* $i \in \mathsf{Hon}$ *and* $j \in \mathsf{Cor}$*, output* $\perp$*. Otherwise return the reencrypted ciphertext* $\mathsf{ReEnc}(\mathsf{rk}_{i\to j}, \mathsf{ct}_i)$.

Challenge Oracle: *On input* $(i, \mathbf{m}_0, \mathbf{m}_1)$ *where* $i \in \mathsf{Hon}$ *and* $\mathbf{m}_0, \mathbf{m}_1 \in \mathcal{M}$*, sample a bit* $b \leftarrow \{0, 1\}$ *uniformly at random, and return the challenge ciphertext* $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \mathbf{m}_b)$*. This oracle can only be queried once.*

Phase 3:

Decision: *On input a bit* $b' \in \{0, 1\}$*, return* win *if* $b = b'$.

The CPA advantage *of* $\mathcal{A}$ *is defined as*

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{cpa}}(\lambda) = \Pr[\mathsf{win}],$$

where the probability is over the randomness of $\mathcal{A}$ *and the oracles in the CPA game.*

Definition 4 (Proxy Reencryption: CPA Security [ABH09]). *Given a security parameter* 1^λ*, a proxy reencryption scheme is CPA secure if for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, there exists a negligible function* negl *such that*

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{cpa}}(\lambda) < \frac{1}{2} + \mathsf{negl}(\lambda)$$

Remark 1. Another definitional subtlety of proxy reencryption worth discussing affects not just CPA security, but HRA and CCA security as well. Consider the specification of $\mathcal{O}_{\mathsf{ReKeyGen}}$ and $\mathcal{O}_{\mathsf{ReEnc}}$ in Definition 3. As defined, the reencryption keys $\mathsf{rk}_{i\to j}$ are *persistent*: the same key is used each time a pair (i, j) is queried. This follows [ABH09, Definition 2.5] and [ABW+13, FL17], though we find our formalization somewhat simpler.

Contrast this with [ABH09, Definition 2.2] in which reencryption keys are *ephemeral*: they are generated afresh each time either oracle is invoked on the same pair (i, j). [BLMR13, PWA+16, CH07] similarly use ephemeral keys in their definitions. In the remaining papers we examined, it was less clear whether reencryption keys were ephemeral or persistent.

Neither definition implies the other; any scheme secure with persistent keys can be modified into one that is insecure with ephemeral keys, and vice-versa. The definitions, however, are not in serious tension; any scheme secure with persistent keys and having deterministic $\mathsf{ReKeyGen}$ is also secure with ephemeral keys, and $\mathsf{ReKeyGen}$ can in general be derandomized using pseudorandom functions. Of course, one can easily define a hybrid notion stronger than both by allowing the adversary to specify for each query whether it wants to use reencryption keys that are new or old.

We adopt the persistent formalization as it better captures 'typical' use. To the best of our knowledge, all claims in this work can be adapted to the ephemeral setting.

Remark 2. The power of the adversary above can be strengthened by allowing *adaptive* corruptions instead of dividing the game into phases. Our definitions of CPA and HRA security follow the convention of [ABH09] primarily because it is most common in the research literature. For an examination of CPA and HRA security in the adaptive setting, see the subsequent work of Fuchsbauer, Kamath, Klein, and Pietrzak [FKKP18]. Adaptive-secure, bidirectional, CCA secure proxy reencryption has been studied in [CH07, NAL15].

3.1 Concatenation Scheme and Trivial Scheme

The weakness of CPA security lies in the specification of $\mathcal{O}_{\mathsf{ReEnc}}$, which does not reencrypt any ciphertexts from honest to corrupt users. Said another way, $\mathcal{O}_{\mathsf{ReEnc}}$ reencrypts between only those pairs keys for which $\mathcal{O}_{\mathsf{ReKeyGen}}$ outputs a reencryption key (rather than returning $\perp$). We now describe two schemes that are CPA secure, but are insecure against a dishonest receiver of reencrypted ciphertexts. In both schemes, a single ciphertext reencrypted from an honest index to a corrupted index enables the decryption of messages encrypted under the honest index's public key.

Concatenation Scheme Let $\mathsf{PRE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReKeyGen}, \mathsf{ReEnc})$ be a CPA-secure proxy reencryption scheme. Define a new scheme by modifying only reencryption and decryption:

$$\mathsf{ReEnc}'(\mathsf{rk}, \mathsf{ct}) := \mathsf{ReEnc}(\mathsf{rk}, \mathsf{ct}) \| \mathsf{rk}$$

$$\mathsf{Dec}'(\mathsf{sk}, \mathsf{ct}) := \begin{cases} \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}^1) \text{ if } \mathsf{ct} = \mathsf{ct}^1 \| \mathsf{ct}^2 \\ \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) \quad \text{otherwise} \end{cases}$$

Theorem 1. *Let* $\mathsf{PRE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReKeyGen}, \mathsf{ReEnc})$ *be a CPA-secure proxy reencryption scheme. The corresponding Concatenation Scheme* $\mathsf{PRE}' = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}', \mathsf{ReKeyGen}, \mathsf{ReEnc}')$ *is a CPA-secure proxy reencryption scheme.*

Proof. For any probabilistic, polynomial-time algorithm $\mathcal{A}'$ (the CPA adversary against PRE'), we construct an efficient algorithm $\mathcal{A}$ such that $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{cpa}} = \mathsf{Adv}^{\mathcal{A}'}_{\mathsf{cpa}}$. By the CPA security of PRE, this advantage is negligible, completing the proof.

$\mathcal{A}$ runs $\mathcal{A}'$ and simulates the CPA security game for PRE' (if $\mathcal{A}'$ does not follow the specification of the game, $\mathcal{A}$ simply aborts). Except for calls to $\mathcal{O}_{\mathsf{ReEnc}}$, all oracle calls by $\mathcal{A}'$ are passed along unaltered by $\mathcal{A}$, along with their responses.

$\mathcal{A}$ begins Phase 2 by requesting all admissible reencryption keys $\mathsf{rk}_{i\to j}$ from its own reencryption key generation oracle. To answer oracle calls from $\mathcal{A}'$ to $\mathcal{O}_{\mathsf{ReEnc}}$, $\mathcal{A}$ first queries its own reencryption oracle, which returns ct^1. If $\mathsf{ct}^1 = \perp$, then $\mathcal{A}'$ returns $\perp$. Otherwise, $\mathcal{A}'$ concatenates the appropriate reencryption key rk to form the new ciphertext $\mathsf{ct} = \mathsf{ct}^1 \| \mathsf{rk}$. This is possible because if $\mathsf{ct}^1 \neq \perp$, then $\mathcal{A}$ is able to get the corresponding reencryption key at the beginning of Phase 2.

$\mathcal{A}$ perfectly implements the CPA security game for PRE', and $\mathcal{A}'$ wins that game if and only if $\mathcal{A}$ wins the corresponding game for PRE. Therefore, $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{cpa}} = \mathsf{Adv}^{\mathcal{A}'}_{\mathsf{cpa}}$. Finally, the running time of $\mathcal{A}$ is polynomially related to that of $\mathcal{A}'$.

While the Concatenation Scheme builds upon any CPA-secure proxy reencryption scheme, the Trivial Scheme presented next makes use of public-key encryption enjoying *circular security*. Informally, circular security guarantees that encryptions of messages that are a function of the secret key(s) are as secure as encryptions of messages that are independent of the secret key(s), a security property that does not follow from standard semantic security.

In the Trivial Scheme, the reencryption key from party i to j is simply $\mathsf{rk}_{i\to j} = \mathsf{Enc}(\mathsf{pk}_j, \mathsf{sk}_i)$. CPA security of the resulting proxy reencryption scheme requires security against an adversary who has both $\mathsf{rk}_{i\to j}$ and $\mathsf{rk}_{j\to i}$. This requires that the underlying encryption scheme is circular secure.

Because existing definitions and constructions of circular secure encryption schemes based on standard assumptions (e.g., [BHHO08] from DDH) require a bound on the total number of public keys n, the corresponding Trivial Scheme will only satisfy a bounded-key variant of CPA security. Any circular secure encryption scheme without this limitation would yield a Trivial Scheme secure according to Definition 4. We defer the definitions of circular security, bounded-key CPA security, and the proof of Theorem 2 to Appendix A.

Trivial Scheme Let $(\mathsf{KeyGen}_{\mathsf{circ}}, \mathsf{Enc}_{\mathsf{circ}}, \mathsf{Dec}_{\mathsf{circ}})$ be an n-way circular-secure encryption scheme. Let $\mathsf{Setup} \equiv \perp$, $\mathsf{KeyGen} \equiv \mathsf{KeyGen}_{\mathsf{circ}}$; $\mathsf{Enc} \equiv \mathsf{Enc}_{\mathsf{circ}}$;

$$\mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j) := \mathsf{Enc}_{\mathsf{circ}}(\mathsf{pk}_j, \mathsf{sk}_i)$$
$$\mathsf{ReEnc}(\mathsf{rk}_{i\to j}, \mathsf{ct}_i) := \mathsf{ct}_i \| \mathsf{rk}_{i\to j}$$
$$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) := \begin{cases} \mathsf{Dec}_{\mathsf{circ}}(\mathsf{Dec}_{\mathsf{circ}}(\mathsf{sk}, \mathsf{ct}^2), \mathsf{ct}^1) & \text{if } \mathsf{ct} = \mathsf{ct}^1 \| \mathsf{ct}^2 \\ \mathsf{Dec}_{\mathsf{circ}}(\mathsf{sk}, \mathsf{ct}) & \text{otherwise} \end{cases}.$$

Theorem 2. *Let* $(\mathsf{KeyGen}_{\mathsf{circ}}, \mathsf{Enc}_{\mathsf{circ}}, \mathsf{Dec}_{\mathsf{circ}})$ *be an* n*-way circular-secure encryption scheme. The corresponding Trivial Scheme* PRE *is an* n*-way CPA secure proxy reencryption scheme.*

4 Security Against Honest Reencryption Attacks

We seek a definition of security which holds as long as the adversary only sees honestly reencrypted ciphertexts, unless the set of corrupt parties can trivially violate security (by some chain of reencryption keys from an uncorrupted public key to a corrupted public key).

We term this notion security against *honest reencryption attacks (HRA)*. To formalize it, we model the ability of an adversary to see honest reencryptions by granting it access to a modified reencryption oracle $\mathcal{O}_{\mathsf{ReEnc}}$. Instead of taking a ciphertext as input, the modified $\mathcal{O}_{\mathsf{ReEnc}}$ takes as input a reference to a previously generated ciphertext (either as the output of $\mathcal{O}_{\mathsf{Enc}}$ or $\mathcal{O}_{\mathsf{ReEnc}}$ itself).

To implement such an oracle, we introduce to the security game a key-value store $\mathcal{C}$ as additional state: the values are ciphertexts ct which are keyed by a pair of integers (i, k), where i denotes the index of the key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$ under which ct was (re)encrypted, and k is a unique index assigned to ct.

As described, this new $\mathcal{O}_{\mathsf{ReEnc}}$ admits a trivial attack: simply reencrypt the challenge ciphertext to a corrupted key and decrypt. To address this issue, we adapt an idea from [CH07]'s definition of CCA security: we rule out the trivial attack by storing a set Deriv of ciphertexts derived from the challenge by reencrypting, and rejecting queries to $\mathcal{O}_{\mathsf{ReEnc}}$ for ciphertexts in Deriv and a corrupted target key. We might have instead chosen to forbid any reencryptions of the challenge ciphertext, even between uncorrupted keys. Though this would have simplified the definition, it would have been unsatisfactory. For example, in the single-writer, many-reader encrypted storage application the contents of a message $\mathbf{m}$ that gets reencrypted from Alice to Charlie should be hidden from Bob.

We now present the honest reencryption attacks security game. The game is similar to the CPA security game with some modifications to Setup, Challenge, and $\mathcal{O}_{\mathsf{ReEnc}}$, and the addition of an Enc oracle $\mathcal{O}_{\mathsf{Enc}}$ to Phase 2. $\mathcal{O}_{\mathsf{Enc}}$ may be executed $\mathrm{poly}(\lambda)$ times and in any order relative to the other oracles in Phase 2. For the sake of clarity we reproduce the full game below and mark the modified oracles with a $\star$.

Definition 5 (Proxy Reencryption: Security Game for Honest Reencryption Attacks (HRA)). *Let λ be the security parameter and $\mathcal{A}$ be an oracle Turing machine. The HRA game consists of an execution of $\mathcal{A}$ with the following oracles.*

Phase 1:

- $\star$ Setup: *The public parameters are generated and given to $\mathcal{A}$. A counter* $\mathsf{numKeys}$ *is initialized to 0, and the sets* Hon *(of honest, uncorrupted indices) and* Cor *(of corrupted indices) are initialized to be empty.*

  Additionally the following are initialized: a counter numCt *to 0, a key-value store $\mathcal{C}$ to empty, and a set* Deriv *to be empty. This oracle is executed first and only once.*

Uncorrupted Key Generation: *Obtain a new key pair* $(\mathsf{pk}_{\mathsf{numKeys}}, \mathsf{sk}_{\mathsf{numKeys}}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ *and give* $\mathsf{pk}_{\mathsf{numKeys}}$ *to* $\mathcal{A}$*. The current value of* numKeys *is added to* Hon *and* numKeys *is incremented.*

Corrupted Key Generation: *Obtain a new key pair* $(pk_{\mathsf{numKeys}}, \mathsf{sk}_{\mathsf{numKeys}}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ *and given to* $\mathcal{A}$*. The current value of* numKeys *is added to* Cor *and* numKeys *is incremented.*

Phase 2: For each pair $i, j \leq \mathsf{numKeys}$*, compute the reencryption key* $\mathsf{rk}_{i \to j} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$.

Reencryption Key Generation $\mathcal{O}_{\mathsf{ReKeyGen}}$: *On input* (i, j) *where* $i, j \leq \mathsf{numKeys}$*, if* $i = j$ *or if* $i \in \mathsf{Hon}$ *and* $j \in \mathsf{Cor}$*, output* $\perp$*. Otherwise return the reencryption key* $\mathsf{rk}_{i \to j}$.

$\star$ Encryption $\mathcal{O}_{\mathsf{Enc}}$: *On input* $(i, \mathbf{m})$*, where* $i \leq \mathsf{numKeys}$*, compute* $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \mathbf{m})$ *and increment* numCt*. Store the value* ct *in* $\mathcal{C}$ *with key* (i, numCt)*. Return* $(\mathsf{numCt}, \mathsf{ct})$.

$\star$ Challenge Oracle: *On input* $(i, \mathbf{m}_0, \mathbf{m}_1)$ *where* $i \in \mathsf{Hon}$ *and* $\mathbf{m}_0, \mathbf{m}_1 \in \mathcal{M}$*, sample a bit* $b \leftarrow \{0, 1\}$ *uniformly at random, compute the challenge ciphertext* $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pk}_i, \mathbf{m}_b)$*, and increment* numCt*. Add* numCt *to the set* Deriv*. Store the value* ct^* *in* $\mathcal{C}$ *with key* (i, numCt)*. Return* $(\mathsf{numCt}, \mathsf{ct}^*)$*. This oracle can only be queried once.*

$\star$ Reencryption $\mathcal{O}_{\mathsf{ReEnc}}$: *On input* (i, j, k) *where* $i, j \leq \mathsf{numKeys}$ *and* $k \leq \mathsf{numCt}$*, if* $j \in \mathsf{Cor}$ *and* $k \in \mathsf{Deriv}$ *return* $\perp$*. If there is no value in* $\mathcal{C}$ *with key* (i, k)*, return* $\perp$*.*

Otherwise, let ct_i *be that value in* $\mathcal{C}$*, let* $\mathsf{ct}_j \leftarrow \mathsf{ReEnc}(\mathsf{rk}_{i \to j}, \mathsf{ct}_i)$*, and increment* numCt*. Store the value* ct_j *in* $\mathcal{C}$ *with key* (j, numCt)*. If* $k \in \mathsf{Deriv}$*, add* numCt *to the set* Deriv.
Return $(\mathsf{numCt}, \mathsf{ct}_j)$.

Phase 3:

Decision: *On input a bit* $b' \in \{0, 1\}$*, return* win *if* $b = b'$.

The HRA advantage *of* $\mathcal{A}$ *is defined as*

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{hra}}(\lambda) = \Pr[\mathsf{win}],$$

where the probability is over the randomness of $\mathcal{A}$ *and the oracles in HRA game.*

Definition 6 (Proxy Reencryption: HRA Security). *Given a security parameter* 1^λ*, a proxy reencryption scheme is HRA secure if for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, there exists a negligible function* negl *such that*

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{hra}}(\lambda) < \frac{1}{2} + \mathsf{negl}(\lambda)$$

The Concatenation Scheme demonstrates that CPA security does not necessarily imply HRA security. Together with following theorem, we see that HRA security is a strict strengthening of CPA security.

Theorem 3. *Let* PRE *be an HRA secure proxy reencryption scheme. Then* PRE *is CPA secure.*

Proof. From any probabilistic, polynomial-time algorithm $\mathcal{A}$ (the CPA adversary), we construct an efficient algorithm $\mathcal{A}'$ such that $\mathsf{Adv}_{\mathsf{hra}}^{\mathcal{A}'} = \mathsf{Adv}_{CPA}^{\mathcal{A}}$. By the HRA security of PRE this advantage is negligible, completing the proof.

$\mathcal{A}'$ runs $\mathcal{A}$ and simulates the CPA security game (if $\mathcal{A}$ does not follow the specification of the CPA security game, $\mathcal{A}'$ simply aborts). Except for calls to $\mathcal{O}_{\mathsf{ReEnc}}$, all oracle calls by $\mathcal{A}'$ are passed along unaltered by $\mathcal{A}$ to the corresponding HRA oracles, along with their responses.

$\mathcal{A}'$ begins Phase 2 by requesting all (admissible) reencryption keys $\mathsf{rk}_{i \to j}$ from $\mathcal{O}_{\mathsf{ReKeyGen}}$. racle calls from $\mathcal{A}$ to $\mathcal{O}_{\mathsf{ReEnc}}$ are answered by $\mathcal{A}'$ by computing the reencryption using the appropriate reencryption key; this is possible because $\mathcal{O}_{\mathsf{ReEnc}}$ returns $\perp$ if and only if $\mathcal{A}'$ is unable to get the corresponding reencryption key.

$\mathcal{A}'$ prefectly implements the CPA security game, and $\mathcal{A}$ wins that game if and only if $\mathcal{A}'$ wins the HRA security game. Therefore $\mathsf{Adv}_{\mathsf{hra}}^{\mathcal{A}'} = \mathsf{Adv}_{CPA}^{\mathcal{A}}$. Finally, the running time of $\mathcal{A}'$ is polynomially related to the that of $\mathcal{A}$.

4.1 Sufficiency of HRA Security for Applications

Returning to the applications of proxy reencryption described in Section 2, we observe that HRA security provides substantially stronger security guarantees.

Encrypted email forwarding Using proxy reencryption with HRA security, Alice can forward encrypted email to Bob for a short period of time (during a vacation, say) and be sure that Bob cannot read her email after she returns.

Key escrow Similar to encrypted email forwarding, proxy reencryption with HRA can be used to enable law enforcement to read certain encrypted messages without compromising the secrecy of documents outside the scope of a search warrant or subpoena, for example.

Single-writer, many-reader encrypted storage Whereas proxy reencryption with CPA security can only support all or nothing access (i.e., all readers may access all data), HRA security can support fine grained access control (i.e., each reader may access only a specific subset of the data).

There is no question that HRA does not provide as much security as CCA, and that CCA-secure proxy reencryption would yield more robust applications. HRA security, however, can provide meaningful guarantees in the above applications.

Encrypted email forwarding If Alice is forwarding all emails to Bob, then Bob could certainly mount an attack outside the honest reencryption model. On the other hand, if Alice is forwarding only those emails from a third-party sender Charlie, then such an attack is impossible without the involvement of Charlie (supposing of course that the sender of an email can be authenticated).

Key escrow The hypothetical legal regime that establishes the government's power of exceptional access by way of key escrow could additionally prohibit the government from mounting chosen-ciphertext attacks. In the United States, a Constitutional argument could perhaps be made that law-enforcement use of chosen-ciphertext attacks must be limited.

Single-writer, many-reader encrypted storage The only ciphertexts being reencrypted are those uploaded by the single-writer to the proxy server (hence the name). It is by no means a stretch to require that the proxy server does not allow writes by unauthorized parties (i.e., the readers). If the honest writer only uploads honestly generated ciphertexts, HRA is appropriate.

5 Security of Existing Proxy Reencryption Schemes

Can we construct HRA-secure proxy reencryption? The most natural place to begin is with existing schemes.

We begin by demonstrating that the CPA secure scheme of [PRSV17] is not HRA secure. Although CPA security is strictly weaker than HRA security, we might hope that the existing CPA secure schemes already satisfy the more stringent definition. To this end, we identify a natural property—*reencryption simulatability*—sufficient to boost CPA security to HRA security.[7]

We examine the simple construction of CPA secure proxy reencryption from any fully-homomorphic encryption (FHE) presented in [Gen09]. While the resulting proxy reencryption may not be HRA secure in general, if the FHE is *circuit private*—a property Gentry imbues into his FHE by rerandomization—the same construction will be HRA secure. We then examine pairing-based schemes, finding there too that rerandomization provides a direct path to HRA security.[8]

Remark 3. It may seem that CCA security for proxy reencryption should imply HRA security, but unfortunately the situation is not so simple. While the relationship between CCA and HRA security is still open, CCA security does imply CPA security. By Theorem 5, any CCA secure proxy reencryption scheme satisfying reencryption simulatability is also HRA secure. See Appendix B for further discussion.

5.1 HRA Insecurity of [PRSV17]

Though it is easy to construct CPA secure encryption schemes that are not HRA secure, the question remains whether any previously proposed schemes fail to

[7] Some existing notions in the proxy reencryption literature seem powerful enough to elevate CPA security to HRA security, including proxy invisibility [AFGH06], unlinkability [FL17], and punctured security [ACJ17]. However, these notions are not sufficiently well defined to draw any concrete conclusions. The notion of key-privacy [ABH09] does not in general suffice for HRA security.

[8] While we do not examine every pairing-based construction of proxy reencryption, we suspect that rerandomizing reencryption will suffice for reencryption simulation in many, if not all.

satisfy HRA security. In this section, we show that the construction of Polyakov, Rohloff, Sahu, and Vaikuntanathan [PRSV17, Section 5] is one such scheme. Their construction is based on a public key encryption scheme of Brakerski and Vaikuntanathan [BV11] and is CPA secure assuming the hardness of Ring Learning With Errors (RLWE).

As with the Trivial and Concatenation schemes, the HRA attack is simple yet severe: any single honestly generated ciphertext enables the recipient Bob to recover the sender Alice's secret key with significant probability.

Theorem 4. *The proxy reencryption scheme of [PRSV17, Section 5] is not HRA secure.*

Proof. Except where noted, the notation used below is consistent with [PRSV17]; we restrict our description to those facts necessary to describe the HRA attack.

For n a power of 2 and prime $q \equiv 1 \mod 2n$, let $R_q = \mathbb{Z}_q[x]/(x^n + 1)$ be a ring of degree $(n-1)$ polynomials with coefficients in $\mathbb{Z}_q$. The sender Alice's secret key is s, and the recipient Bob's secret key is s^*. Bob's public key includes $O(\log q) = \text{poly}(n)$ RLWE samples $\theta_i^* = \beta_i \cdot s^* + pe_i$, where p is a public prime and the β_i and e_i are ring elements sampled by Bob.[9] Ciphertexts are pairs of ring elements $(c_0, c_1) \in R_q^2$. By [BV11, Lemma 4], the distribution of c_1 is statistically close to uniform over R_q. By [LPR13, Lemma 2.25], c_1 is invertible with probability at least $e^{-1} - \mathsf{negl}(n)$. The result of reencrypting (c_0, c_1) is a pair (c_0', c_1') such that $c_0' - s^* \cdot c_1' = c_0 - s \cdot c_1 + pE_1$, where E_i is computable given c_1 and the e_i. This fact is used by [PRSV17] to prove the correctness of their scheme. Rearranging the above, we see that

$$s \cdot c_1 = c_0 + pE_1 - c_0' + s^* \cdot c_1'.$$

Given any ciphertext (c_0, c_1) and its reencryption (c_0', c_1'), Bob can evaluate the above and compute $s \cdot c_1$. With probability at least $e^{-1} - \mathsf{negl}(n)$, c_1 is invertible and Bob can recover the secret s.

5.2 Reencryption Simulatability.

While HRA is a strictly stronger security notion than CPA, we now show that if a CPA secure proxy reencryption scheme has an additional property we call *reencryption simulatability*, then it must also be HRA secure. Very roughly, reencryption simulatability means that ciphertexts resulting from computing $\mathsf{ReEnc}(\mathsf{rk}_{i\to j}, \mathsf{ct}_i)$ can be simulated without knowledge of the sender's secret key sk_i (but with knowledge of the plaintext message $\mathbf{m}$ and the recipient's secret key sk_j). Note that ReEnc uses $\mathsf{rk}_{i\to j}$ which is a function of sk_i.

Reencryption simulatability allows an algorithm with access to the CPA oracles to efficiently implement the honest reencryption oracle. For intuition, consider the following approach to reducing HRA security to CPA security. Suppose

[9] [PRSV17] separate the computation of θ_i^* from Bob's public key, but this is only a matter of presentation.

there existed an adversary $\mathcal{A}_{\mathsf{hra}}$ violating the HRA security of a scheme; the reduction plays the roles of both the CPA adversary and the challenger in the HRA security game, and attempts to violate CPA security. To succeed, the reduction must be able to answer honest reencryption queries from uncorrupted keys to corrupted keys. Though the reduction knows the messages being reencrypted, it does not know the appropriate reencryption key. However, if it could indistinguishably simulate these reencryptions then it could indeed leverage $\mathcal{A}_{\mathsf{hra}}$ to violate CPA security.

We emphasize that the goal of Definition 7 is to capture a large swath of possible schemes while still enabling very simple proofs of simulatability (and thus of HRA security for existing CPA secure schemes). It is not intended to be the only avenue for proving HRA security of new or existing constructions. Reencryption simulatability is not necessary for HRA security of proxy reencryption. In particular, analogous versions of Theorem 5 hold with computational simulatability guarantees, but are more complicated [DN18, Foonote 7 and Appendix A].

Definition 7 (Reencryption Simulatability). *A proxy reencryption scheme* PRE *is* reencryption simulatable *if there exists a probabilistic, polynomial-time algorithm* ReEncSim *such that with high probability over* aux*, for all* $\mathbf{m} \in \mathcal{M}$*:*

$$(\mathsf{ReEncSim}(\mathsf{aux}), \mathsf{aux}) \approx_s (\mathsf{ReEnc}(\mathsf{rk}_{a\to b}, \mathsf{ct}_a), \mathsf{aux}),$$

where $\approx_s$ *denotes statistical indistinguishability, and* ct_a *and* aux *are sampled according to*

$$\begin{aligned}
&\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda),\\
&(\mathsf{pk}_a, \mathsf{sk}_a) \leftarrow \mathsf{KeyGen}(\mathsf{pp}),\\
&(\mathsf{pk}_b, \mathsf{sk}_b) \leftarrow \mathsf{KeyGen}(\mathsf{pp}),\\
&\mathsf{rk}_{a\to b} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_a, \mathsf{pk}_b),\\
&\mathsf{ct}_a \leftarrow \mathsf{Enc}(\mathsf{pk}_a, \mathbf{m}),\\
&\mathsf{aux} = (\mathsf{pp}, \mathsf{pk}_a, \mathsf{pk}_b, \mathsf{sk}_b, \mathsf{ct}_a, \mathbf{m}).
\end{aligned}$$

A special case of the above is when $\mathsf{ReEncSim}(\mathsf{aux}) = \mathsf{Enc}(\mathsf{pk}_b, \mathbf{m})$ simply computes a fresh encryption of the message. That is, if reencrypted ciphertexts are distributed like fresh ciphertexts, then the scheme is reencryption simulatable.

Theorem 5. *Let* PRE *be an CPA secure, reencryption simulatable, proxy reencryption scheme. Then* PRE *is HRA secure.*

Proof (Outline). The proof proceeds according to the intuition above. From any probabilistic, polynomial-time algorithm $\mathcal{A} = \mathcal{A}_{\mathsf{hra}}$ (the HRA adversary), we construct an algorithm $\mathcal{A}' = \mathcal{A}_{\mathsf{cpa}}$ such that $\mathsf{Adv}^{\mathcal{A}'}_{\mathsf{cpa}}(\lambda) \geq \mathsf{Adv}^{\mathcal{A}}_{\mathsf{hra}}(\lambda) - \mathsf{negl}(\lambda)$; by the CPA security of PRE this advantage is negligible, completing the proof.

$\mathcal{A}_{\mathsf{cpa}}$ runs $\mathcal{A}_{\mathsf{hra}}$ and simulates the HRA security game (if $\mathcal{A}_{\mathsf{hra}}$ does not follow the specification of the HRA security game, $\mathcal{A}_{\mathsf{cpa}}$ simply aborts). To answer

oracle calls by $\mathcal{A}_{\mathsf{hra}}$ to any oracle other than $\mathcal{O}_{\mathsf{ReEnc}}$, $\mathcal{A}_{\mathsf{cpa}}$ simply passes the calls and answers unaltered to the corresponding CPA oracles.

To answer oracle calls to $\mathcal{O}_{\mathsf{ReEnc}}$ between two uncorrupted keys or two corrupted keys, $\mathcal{A}_{\mathsf{cpa}}$ uses the corresponding reencryption key. On the other hand, for calls to $\mathcal{O}_{\mathsf{ReEnc}}$ from an uncorrupted key to a corrupted key, $\mathcal{A}_{\mathsf{cpa}}$ simulates the reencryption using $\mathsf{ReEncSim}$. This is possible because $\mathcal{A}_{\mathsf{cpa}}$ knows the underlying $\mathbf{m}$ (along with the other information in aux).

Reencryption simulatability implies that the views of $\mathcal{A}_{\mathsf{hra}}$ in the real security game (using the real $\mathcal{O}_{\mathsf{ReEnc}}$) and the simulated security game (using $\mathsf{ReEncSim}$) are statistically close. $\mathcal{A}_{\mathsf{cpa}}$ wins the CPA security game if and only if $\mathcal{A}_{\mathsf{hra}}$ wins in the simulated HRA game described above.

5.3 Fully Homomorphic Encryption and Proxy Reencryption

There is an intimate connection between FHE and proxy reencryption: a sufficiently powerful somewhat homomorphic encryption scheme implies CPA secure proxy reencryption, which can be used to "bootstrap" the scheme to achieve fully homomorphic encryption [Gen09]. For the relevant FHE definitions, see [Gen09, Section 2].

Let $\mathsf{FHE} = (\mathsf{Setup}_{\mathsf{FHE}}, \mathsf{KeyGen}_{\mathsf{FHE}}, \mathsf{Enc}_{\mathsf{FHE}}, \mathsf{Dec}_{\mathsf{FHE}}, \mathsf{Eval}_{\mathsf{FHE}})$ be an FHE scheme. Proxy reencryption can be constructed as follows (compare to the Trivial Scheme):

$\mathsf{KeyGen}_{\mathsf{PRE}}$, $\mathsf{Enc}_{\mathsf{PRE}}$ and $\mathsf{Dec}_{\mathsf{PRE}}$ are identical to their FHE counterparts.
$\mathsf{ReKeyGen}_{\mathsf{PRE}}(\mathsf{sk}_i, \mathsf{pk}_j) = \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_j, \mathsf{sk}_i) \| \mathsf{pk}_j$. The reencryption key $\mathsf{rk}_{i \to j}$ is an encryption under pk_j of sk_i, along with the target public key pk_j.
$\mathsf{ReEnc}_{\mathsf{PRE}}(\mathsf{rk}_{i \to j}, \mathsf{ct}_i)$: Let $\mathsf{ct}_{i \to j} \leftarrow \mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}_j, \mathsf{ct}_i)$. Homomorphically compute the FHE decryption function $\mathsf{Dec}_{\mathsf{FHE}}(\mathsf{sk}_i, \mathsf{ct}_i)$ using the corresponding ciphertexts $\mathsf{rk}_{i \to j}$ and $\mathsf{ct}_{i \to j}$ (under pk_j). Namely, $\mathsf{ct}_j = \mathsf{Eval}_{\mathsf{FHE}}(\mathsf{pk}_j, \mathsf{Dec}_{\mathsf{FHE}}, \mathsf{rk}_{i \to j}, \mathsf{ct}_{i \to j})$.

The correctness of the FHE implies the correctness of the resulting proxy reencryption:

$$\mathsf{Dec}_{\mathsf{PRE}}(\mathsf{sk}_j, \mathsf{ct}_j) = \mathsf{Dec}_{\mathsf{FHE}}(\mathsf{sk}_j, \mathsf{ct}_j) = \mathsf{Dec}_{\mathsf{FHE}}(\mathsf{sk}_i, \mathsf{ct}_i) = \mathsf{Dec}_{\mathsf{PRE}}(\mathsf{sk}_i, \mathsf{ct}_i).$$

Furthermore, the proxy reencryption scheme is CPA secure.

To demonstrate that the construction might not be HRA secure, consider the following fully homomorphic encryption scheme FHE' constructed from any existing scheme FHE. The only modification made in FHE' is to $\mathsf{Eval}_{\mathsf{FHE}'}$:

$$\mathsf{Eval}_{\mathsf{FHE}'}(\mathsf{pk}_j, C, \mathsf{ct}_1, \mathsf{ct}_2) := \mathsf{Eval}_{\mathsf{FHE}}(\mathsf{pk}_j, C, \mathsf{ct}_1, \mathsf{ct}_2) \| \mathsf{ct}_1.$$

Note that FHE' does not violate FHE compactness if ct_1 (in the proxy reencryption construction, $\mathsf{rk}_{i \to j}$) is always of some size bounded by a polynomial in the security parameter of the FHE scheme; this suffices for our purpose. Instantiating the proxy reencryption construction with FHE' essentially results in the Concatenation Scheme, which is not HRA secure.

Circuit Privacy. An FHE scheme is *circuit private* if ciphertexts resulting from FHE evaluations are statistically indistinguishable from fresh ciphertexts [Gen09]. Namely, if for any circuit C and any ciphertexts $\mathsf{ct}_1,\ldots,\mathsf{ct}_t$:

$$\mathsf{Enc}_{\mathsf{FHE}}(\mathsf{pk}, C(\mathsf{ct}_1,\ldots,\mathsf{ct}_t)) \approx_s \mathsf{Eval}_{\mathsf{FHE}}(\mathsf{pk}, C, \mathsf{ct}_1,\ldots,\mathsf{ct}_t).$$

In [Gen09], an FHE scheme without circuit privacy is modified to be circuit private by rerandomizing the ciphertexts resulting from $\mathsf{Eval}_{\mathsf{FHE}}$.

While our proxy reencryption construction above is not in general HRA secure, it is easy to see that if the underlying FHE is circuit private, then the proxy reencryption is reencryption simulatable (Definition 7). By Theorem 5, the resulting scheme is therefore HRA secure.

5.4 Pairing-Based Proxy Reencryption

Many constructions of proxy reencryption are based on the hardness of Diffie-Hellman-type problems over certain bilinear groups, including [AFGH06, CH07, LV08, ABH09, HRSV07].

A prototypical construction is that of [AFGH06], which itself is based on the original scheme of [BBS98]. For every λ, let G_1 and G_2 be groups of prime order $q = \Theta(2^\lambda)$, and let g be a generator of G_1. Let e be a non-degenerate bilinear map $e : G_1 \times G_1 \to G_2$ (i.e., for all $h \in G_1$ and $a, b \in \mathbb{Z}_q$, $e(h^a, h^b) = e(h, h)^{ab}$, and for all generators g of G_1, $e(g, g) \neq 1$). Let $Z = e(g, g)$. The message-space of the scheme is G_2.

$\mathsf{Setup}(1^\lambda)$: Output $\mathsf{pp} = (q, g, G_1, G_2, e)$.
$\mathsf{KeyGen}(\mathsf{pp})$: Sample $a \leftarrow \mathbb{Z}_q$ uniformly at random. Output $\mathsf{sk} = a$ and $\mathsf{pk} = g^a$.
$\mathsf{Enc}(\mathsf{pk}, \mathbf{m})$: Sample $k \leftarrow \mathbb{Z}_q$ uniformly at random. Output $\mathsf{ct} = (\mathsf{pk}^k, \mathbf{m}Z^k) = (g^{ak}, \mathbf{m}Z^k)$.
$\mathsf{ReKeyGen}(\mathsf{sk}_A = a, \mathsf{pk}_B = g^b)$: Output $\mathsf{rk}_{A\to B} = g^{b/a}$.
$\mathsf{ReEnc}(\mathsf{rk}_{A\to B}, \mathsf{ct}_A)$: Let $\mathsf{ct}_A = (\alpha_1, \alpha_2)$. Output

$$\mathsf{ct}_B = (e(\alpha_1, \mathsf{rk}_{A\to B}), \alpha_2) = (e(g^{ak}, g^{b/a}), \mathbf{m}Z^k) = (Z^{bk}, \mathbf{m}Z^k).$$

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: Let $\mathsf{ct} = (\alpha_1, \alpha_2)$. If $\alpha_1 \in G_2$ (i.e., if it is the result of ReEnc), then output $\alpha_2/\alpha_1^{1/a} = \mathbf{m}Z^k/Z^k = \mathbf{m}$. Otherwise $\alpha_1 \in G_1$ (i.e., it is a fresh ciphertext); output $\alpha_2/e(\alpha_1, g)^{1/a} = \mathbf{m}Z^k/e(g^{ak}, g)^{1/a} = \mathbf{m}Z^k/Z^k = \mathbf{m}$.

Is this scheme HRA secure? It is tempting to say that the scheme is reencryption simulatable; after all, given a message $\mathbf{m}$ it is indeed straightforward to sample $(Z^{bk}, \mathbf{m}Z^k)$ for random $k \leftarrow \mathbb{Z}_q$. However $\mathsf{ct}_A = (g^{ak}, \mathbf{m}Z^k)$ and $\mathsf{ct}_B = \mathsf{ReEnc}(\mathsf{rk}_{A\to B}, \mathsf{ct}_A) = (Z^{bk}, \mathbf{m}Z^k)$ share the randomness k. Given $\mathsf{ct}_A = (g^{ak}, \mathbf{m}Z^k)$ and $\mathbf{m}$, it is not clear how to output $(g^{bk}, \mathbf{m}Z^k)$.

Rerandomization A minor modification to the scheme above guarantees reencryption simulatability and therefore HRA security. Replace ReEnc above with ReEnc':

$\mathsf{ReEnc}'(\mathsf{rk}_{A\to B}, \mathsf{ct}_A)$: Compute $(Z^{bk}, \mathbf{m}Z^k) = \mathsf{ReEnc}(\mathsf{rk}_{A\to B}, \mathsf{ct}_A)$. Sample $k' \leftarrow \mathbb{Z}$ uniformly at random, and output $(Z^{bk} \cdot e(g^b, g^{k'}), \mathbf{m}Z^k \cdot e(g, g^{k'})) = (Z^{b(k+k')}, \mathbf{m}Z^{k+k'})$.

The resulting proxy reencryption scheme maintains the CPA security of the original, as the only modification is the rerandomization of reencrypted ciphertexts (which can be done by anyone with knowledge of the public parameters).

Furthermore, the scheme is now reencryption simulatable. To see why, observe that the resulting reencrypted ciphertexts are uniformly distributed in $\{(\mathsf{ct}_1, \mathsf{ct}_2) \in G_2 \times G_2 : \mathsf{ct}_2/\mathsf{ct}_1^{1/b} = \mathbf{m}\}$, independent of all other keys and ciphertexts. Such ciphertexts are easily sampled with knowledge of pp, $\mathsf{pk}_B = g^b$ and $\mathbf{m}$ as follows.

$\mathsf{ReEncSim}(\mathsf{pp}, \mathsf{pk}_B, \mathbf{m})$: Sample $k' \leftarrow \mathbb{Z}_q$ uniformly at random, and and output $(e(\mathsf{pk}_B, g^{k'}), \mathbf{m} \cdot e(g, g^{k'})) = (Z^{bk'}, \mathbf{m}Z^{bk'})$.

Thus, by Theorem 5, the modified scheme is HRA secure. Observe that rerandomization was the key to achieving circuit privacy (and thereby HRA security) in the FHE-based proxy reencryption construction as well.

The pairing-based schemes of [ABH09] and [HRSV07] already incorporate rerandomization during reencryption. In the former case, it is used to achieve "key privacy;" in the latter, to achieve obfuscation of the reencryption functionality. In each, it is straightforward to show that the schemes are also reencryption simulatable and therefore HRA secure.

References

ABH09. Giuseppe Ateniese, Karyn Benson, and Susan Hohenberger. Key-private proxy re-encryption. In *CT-RSA*, volume 5473, pages 279–294. Springer, 2009.

ABW+13. Yoshinori Aono, Xavier Boyen, Lihua Wang, et al. Key-private proxy reencryption under lwe. In *International Conference on Cryptology in India*, pages 1–18. Springer, 2013.

ACJ17. Prabhanjan Ananth, Aloni Cohen, and Abhishek Jain. Cryptography with updates. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 445–472. Springer, 2017.

AFGH06. Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Transactions on Information and System Security (TISSEC)*, 9(1):1–30, 2006.

BBS98. Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In *Advances in Cryptology—EUROCRYPT'98*, pages 127–144. Springer, 1998.

BHHO08. Dan Boneh, Shai Halevi, Mike Hamburg, and Rafail Ostrovsky. Circular-secure encryption from decision diffie-hellman. In *Annual International Cryptology Conference*, pages 108–125. Springer, 2008.

BLMR13. Dan Boneh, Kevin Lewi, Hart Montgomery, and Ananth Raghunathan. Key homomorphic prfs and their applications. In *Advances in Cryptology–CRYPTO 2013*, pages 410–428. Springer, 2013.

BPR+17. Cristian Borcea, Yuriy Polyakov, Kurt Rohloff, Gerard Ryan, et al. Picador: End-to-end encrypted publish–subscribe information distribution with proxy re-encryption. *Future Generation Computer Systems*, 71:177–191, 2017.

BV11. Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In *Annual cryptology conference*, pages 505–524. Springer, 2011.

CCL+14. Nishanth Chandran, Melissa Chase, Feng-Hao Liu, Ryo Nishimaki, and Keita Xagawa. Re-encryption, functional re-encryption, and multi-hop re-encryption: A framework for achieving obfuscation-based security and instantiations from lattices. In *PKC*, pages 95–112. Springer, 2014.

CH07. Ran Canetti and Susan Hohenberger. Chosen-ciphertext secure proxy re-encryption. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 185–194. ACM, 2007.

CKN03. Ran Canetti, Hugo Krawczyk, and Jesper B Nielsen. Relaxing chosen-ciphertext security. In *Annual International Cryptology Conference*, pages 565–582. Springer, 2003.

CWYD10. Sherman SM Chow, Jian Weng, Yanjiang Yang, and Robert H Deng. Efficient unidirectional proxy re-encryption. In *International Conference on Cryptology in Africa*, pages 316–332. Springer, 2010.

DKL+18. David Derler, Stephan Krenn, Thomas Lorünser, Sebastian Ramacher, Daniel Slamanig, and Christoph Striecks. Revisiting proxy re-encryption: Forward secrecy, improved security, and applications. In *IACR International Workshop on Public Key Cryptography*, pages 219–250. Springer, 2018.

DN18. Nico Döttling and Ryo Nishimaki. Universal proxy re-encryption. Cryptology ePrint Archive, Report 2018/840, 2018. https://eprint.iacr.org/2018/840.

EPRS17. Adam Everspaugh, Kenneth Paterson, Thomas Ristenpart, and Sam Scott. Key rotation for authenticated encryption. In *Annual International Cryptology Conference*, pages 98–129. Springer, 2017.

FKKP18. Georg Fuchsbauer, Chethan Kamath, Karen Klein, and Krzysztof Pietrzak. Adaptively secure proxy re-encryption. Cryptology ePrint Archive, Report 2018/426, 2018. https://eprint.iacr.org/2018/426.

FL17. Xiong Fan and Feng-Hao Liu. Proxy re-encryption and re-signatures from lattices. 2017.

Gen09. Craig Gentry. *A fully homomorphic encryption scheme*. Stanford University, 2009.

HHY11. Yi-Jun He, Lucas CK Hui, and Siu Ming Yiu. Avoid illegal encrypted drm content sharing with non-transferable re-encryption. In *Communication Technology (ICCT), 2011 IEEE 13th International Conference on*, pages 703–708. IEEE, 2011.

HRSV07. Susan Hohenberger, Guy Rothblum, Abhi Shelat, and Vinod Vaikuntanathan. Securely obfuscating re-encryption. *Theory of Cryptography*, pages 233–252, 2007.

ID03. Anca-Andreea Ivan and Yevgeniy Dodis. Proxy cryptography revisited. In *NDSS*, 2003.

Jak99. Markus Jakobsson. On quorum controlled asymmetric proxy re-encryption. In *International Workshop on Public Key Cryptography*, pages 112–121. Springer, 1999.

KHP06. Himanshu Khurana, Jin Heo, and Meenal Pant. From proxy encryption primitives to a deployable secure-mailing-list solution. In *International Conference on Information and Communications Security*, pages 260–281. Springer, 2006.

LPK10. Sangho Lee, Heejin Park, and Jong Kim. A secure and mutual-profitable drm interoperability scheme. In *Computers and Communications (ISCC), 2010 IEEE Symposium on*, pages 75–80. IEEE, 2010.

LPR13. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-lwe cryptography. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 35–54. Springer, 2013.

LV08. Benoît Libert and Damien Vergnaud. Unidirectional chosen-ciphertext secure proxy re-encryption. In *International Workshop on Public Key Cryptography*, pages 360–379. Springer, 2008.

NAL15. David Nunez, Isaac Agudo, and Javier Lopez. A parametric family of attack models for proxy re-encryption. In *Computer Security Foundations Symposium (CSF), 2015 IEEE 28th*, pages 290–301. IEEE, 2015.

OMD91. Frank Oz, Bill Murray, and Richard Dreyfuss. *What About Bob*. Touchstone Pictures, 1991.

PRSV17. Yuriy Polyakov, Kurt Rohloff, Gyana Sahu, and Vinod Vaikuntanathan. Fast proxy re-encryption for publish/subscribe systems. *ACM Transactions on Privacy and Security (TOPS)*, 20(4):14, 2017.

PWA+16. L Phong, L Wang, Y Aono, M Nguyen, and X Boyen. Proxy re-encryption schemes with key privacy from lwe. Technical report, Cryptology ePrint Archive, Report 2016/327, 2016. http://eprint. iacr. org/2016/327, 2016.

A The Trivial Scheme

The following description and definition of circular security is adapted with slight modification from [BHHO08].

Let $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme with key space $\mathcal{K}$ and message space $\mathcal{M}$ such that $\mathcal{K} \subseteq \mathcal{M}$. Let $n > 0$ be an integer and let $\mathcal{C}$ be the set of functions $\mathcal{C} = \{f : \mathcal{K}^n \to \mathcal{M}\}$ consisting of

- all $|\mathcal{M}|$ constant functions $f_m(z) = m$ for all $z \in \mathcal{K}^n$, and
- all n selector functions $f_i(x_1, \ldots, x_n) = x_i$ for $1 \leq i \leq n$.

We define circular security using the following game between a challenger and an adversary $\mathcal{A}$. For an integer $n > 0$ and a security parameter λ, the game proceeds as follows:

Initialization: The challenger chooses a random bit $b \leftarrow \{0, 1\}$. It generates $(\mathsf{pk}_1, \mathsf{sk}_1), \ldots, (\mathsf{pk}_n, \mathsf{sk}_n)$ by running $\mathsf{KeyGen}(1^\lambda)$ n times, and sends $(\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ to $\mathcal{A}$.

Queries: The adversary repeatedly issues queries where each query is of the form (i, f) with $1 \leq i \leq n$ and $f \in \mathcal{C}$. The challenger responds by setting $y = f(\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$ and

$$\mathsf{ct} \leftarrow \begin{cases} \mathsf{Enc}(\mathsf{pk}_i, y) & \text{if } b = 0 \\ \mathsf{Enc}(\mathsf{pk}_i, 0^{|y|}) & \text{if } b = 1 \end{cases}$$

and sends ct to $\mathcal{A}$.

Finish: Finally, the adversary outputs a bit $b' \in \{0,1\}$.

We say that $\mathcal{A}$ wins the game if $b = b'$. Let win be the event that $\mathcal{A}$ wins the game and define $\mathcal{A}$'s advantage as

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{circ},n}(\lambda) = \Pr[\mathsf{win}].$$

Definition 8 (n-Circular Security). *We say that a public-key encryption scheme* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is* n-way circular secure *if for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, there exists a negligible function* negl *such that*

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{circ},n}(\lambda) < \frac{1}{2} + \mathsf{negl}(\lambda).$$

Because existing constructions of circularly secure encryption schemes based on standard assumptions require a bound on the total number of public keys n, the corresponding Trivial Scheme will only satisfy a bounded-key variant of CPA security, defined next.

Definition 9 (Proxy Reencryption: n-CPA Security). *For* $n \in \mathbb{N}$*, the* n*-CPA security game is identical to the CPA security game in Definition 3 except for the following underlined modifications. Recall that* $\mathsf{numKeys}$ *is initialized to 0 and is incremented after every key generation call in the security game.*

Uncorrupted Key Generation: *If* $\mathsf{numKeys} = n$*, return* $\perp$*. Otherwise, obtain a new key pair* $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$*.* $\mathcal{A}$ *is given* pk_i*. The current value of* $\mathsf{numKeys}$ *is added to* Hon *and* $\mathsf{numKeys}$ *is incremented.*

Corrupted Key Generation: *If* $\mathsf{numKeys} = n$*, return* $\perp$*. Otherwise, obtain a new key pair* $(pk_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$*.* $\mathcal{A}$ *is given* $(\mathsf{pk}_i, \mathsf{sk}_i)$*. The current value of* $\mathsf{numKeys}$ *is added to* Cor *and* $\mathsf{numKeys}$ *is incremented.*

The corresponding n-CPA advantage *of* $\mathcal{A}$ *is denoted* $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{cpa},n}(\lambda)$*. A proxy reencryption scheme is* n-CPA secure *if for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, there exists a negligible function* negl *such that*

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{cpa},n}(\lambda) < \frac{1}{2} + \mathsf{negl}(\lambda)$$

Trivial Scheme Let $(\mathsf{KeyGen}_{\mathsf{circ}}, \mathsf{Enc}_{\mathsf{circ}}, \mathsf{Dec}_{\mathsf{circ}})$ be an n-way circular secure encryption scheme. Let $\mathsf{Setup} \equiv \perp$, $\mathsf{KeyGen} \equiv \mathsf{KeyGen}_{\mathsf{circ}}$; $\mathsf{Enc} \equiv \mathsf{Enc}_{\mathsf{circ}}$;

$$\mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j) := \mathsf{Enc}_{\mathsf{circ}}(\mathsf{pk}_j, \mathsf{sk}_i)$$

$$\mathsf{ReEnc}(\mathsf{rk}_{i\to j}, \mathsf{ct}_i) := \mathsf{ct}_i \| \mathsf{rk}_{i\to j}$$

$$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) := \begin{cases} \mathsf{Dec}_{\mathsf{circ}}(\mathsf{Dec}_{\mathsf{circ}}(\mathsf{sk}, \mathsf{ct}^2), \mathsf{ct}^1) & \text{if } \mathsf{ct} = \mathsf{ct}^1 \| \mathsf{ct}^2 \\ \mathsf{Dec}_{\mathsf{circ}}(\mathsf{sk}, \mathsf{ct}) & \text{otherwise} \end{cases}.$$

Theorem 2 states that if $(\mathsf{KeyGen}_{\mathsf{circ}}, \mathsf{Enc}_{\mathsf{circ}}, \mathsf{Dec}_{\mathsf{circ}})$ is an n-way circular secure encryption scheme, then the corresponding Trivial Scheme $\mathsf{PRE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReKeyGen}, \mathsf{ReEnc})$ is an n-CPA secure proxy reencryption scheme. In fact, the proof below extends the case when there are n uncorrupted keys and any number of corrupted keys.

Proof (of Theorem 2). For all $n \in \mathbb{N}$ and any probabilistic, polynomial-time algorithm $\mathcal{A}$ (the adversary against the trivial scheme), we construct an efficient algorithm $\mathcal{A}_{\mathsf{circ}}$ such that $\mathsf{Adv}^{\mathcal{A}_{\mathsf{circ}}}_{\mathsf{circ},n} = \frac{1}{2} \cdot \mathsf{Adv}^{\mathcal{A}}_{\mathsf{cpa},n}$. By the hypothesis, this advantage is negligible, completing the proof.

At the beginning of the game, the circular security challenger picks a random bit b. If $b = 0$, then the Queries oracle encrypts all messages correctly; if $b = 1$, then the Queries oracle encrypts only the message 0. $\mathcal{A}_{\mathsf{circ}}$ runs $\mathcal{A}$ and simulates the CPA security game for PRE (if $\mathcal{A}$ does not follow the specification of the game, $\mathcal{A}_{\mathsf{circ}}$ simply aborts).

At the start of Phase 1, $\mathcal{A}_{\mathsf{circ}}$ calls its Initialization oracle in the circular security game. In return it receives the public keys $(\mathsf{pk}_1^{\mathsf{circ}}, \ldots, \mathsf{pk}_n^{\mathsf{circ}})$. To answer an Uncorrupted Key Generation query, $\mathcal{A}_{\mathsf{circ}}$ gives to $\mathcal{A}$ the first unused public key $\mathsf{pk}_i^{\mathsf{circ}}$ from this list. To answer a Corrupted Key Generation query, $\mathcal{A}_{\mathsf{circ}}$ generates a new key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}$ and gives $(\mathsf{pk}, \mathsf{sk})$ to the adversary.

$\mathcal{A}$ begins Phase 2 by using its Queries oracle to learn the reencryption keys for all pairs of uncorrupted keys generated. Using its knowledge of the corrupted secret keys, it also computes reencryption keys for all the pairs of corrupted keys generated. Oracle calls from $\mathcal{A}$ to $\mathcal{O}_{\mathsf{ReKeyGen}}$ are answered with the corresponding reencryption key (or with $\perp$). To answer oracle calls from $\mathcal{A}$ to $\mathcal{O}_{\mathsf{ReEnc}}$, computes the appropriate response; namely, it concatenates the reencryption key to the ciphertext (or returns $\perp$).

At some point, $\mathcal{A}$ queries the Challenge oracle with an honest key index i and a pair of messages $(\mathbf{m}_0, \mathbf{m}_1)$. $\mathcal{A}_{\mathsf{circ}}$ chooses a random one of the messages $\mathbf{m}$ and queries its own Queries oracle with the pair $(i, \mathbf{m})$, returning the resulting ciphertext to $\mathcal{A}$.

Finally, $\mathcal{A}$ guesses whether $\mathbf{m} = \mathbf{m}_0$ or $\mathbf{m}_1$. If $\mathcal{A}$ guesses correctly, $\mathcal{A}_{\mathsf{circ}}$ guesses the bit $b' = 0$. Otherwise, $\mathcal{A}_{\mathsf{circ}}$ guesses a random $b' \leftarrow \{0, 1\}$. Conditioned on $b = 0$, $\mathcal{A}_{\mathsf{circ}}$ perfectly simulates the PRE security game, and guesses $b' = 0$ with probability $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{cpa},n}$. It follows that $\mathsf{Adv}^{\mathcal{A}_{\mathsf{circ}}}_{\mathsf{circ},n} = \frac{1}{2} \cdot \mathsf{Adv}^{\mathcal{A}}_{\mathsf{cpa},n}$.

B CCA Security

It may seem that CCA security for proxy reencryption should imply HRA security, but the situation is not so simple. In this section, we define CCA security for proxy reencryption and describe the challenge in proving that CCA security implies HRA security. Finally, we construct a proxy reencryption scheme that illustrates the problem with the intuition which motivated the original CPA definition which also separates the security models $\mathsf{IND\text{-}CCA}_{0,1}$ and $\mathsf{IND\text{-}CCA}_{2,0}$ defined in [NAL15], proving a converse to their Theorem 4.6.

B.1 Definition

The definition below is adapted from [CH07, Definition 2.4], but modified to simplify comparison to the other definitions presented in this work. First, while [CH07] focuses on bidirectional PRE, we consider unidirectional PRE. Secondly,

we modify the definition to use persistent, rather than ephemeral, reencryption keys (see Remark 1). Finally, we add an intialization stage Setup and generally adapt the syntax to coincide with the notation used throughout this work.[10]

The core concept in the definition is that of *derivatives* of the challenge. Informally, a pair (i, ct) is a derivative of the challenge if the decryption $\mathsf{Dec}(\mathsf{sk}_j, \mathsf{ct})$ or the reencryption $\mathsf{ReEnc}(\mathsf{rk}_{i\to j}, \mathsf{ct})$ to some corrupted key index j would give the adversary "illegitimate information" about the challenge ciphertext. The precise formalization (Definition 11) is reminiscent of *replayable* CCA security for standard encryption [CKN03].

Definition 10 (Proxy Reencryption: Security Game for Chosen Ciphertext Attacks (CCA) [CH07]). *Let λ be the security parameter and $\mathcal{A}$ be an oracle Turing machine. The HRA game consists of an execution of $\mathcal{A}$ with the following oracles, which can be invoked multiple times in any order, subject to the constraints below:*

- Setup: *The public parameters are generated and given to $\mathcal{A}$. A counter* $\mathsf{numKeys}$ *is initialized to 0, and the sets* Hon *(of honest / uncorrupted indices) and* Cor *(of corrupted indices) are initialized to be empty. This oracle is executed first and only once.*
- Challenge Oracle: *On input $(i^*, \mathbf{m}_0, \mathbf{m}_1)$ where $i^* \in \mathsf{Hon}$ and $\mathbf{m}_0, \mathbf{m}_1 \in \mathcal{M}$, sample a bit $b \leftarrow \{0,1\}$ uniformly at random, compute the challenge ciphertext $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{pk}_{i^*}, \mathbf{m}_b)$. Return ct^*. This oracle can only be queried once.*
- Uncorrupted Key Generation: *Obtain a new key pair $(\mathsf{pk}_{\mathsf{numKeys}}, \mathsf{sk}_{\mathsf{numKeys}}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ and give $\mathsf{pk}_{\mathsf{numKeys}}$ to $\mathcal{A}$. The current value of* $\mathsf{numKeys}$ *is added to* Hon *and* $\mathsf{numKeys}$ *is incremented.*
- Corrupted Key Generation: *Obtain a new key pair $(pk_{\mathsf{numKeys}}, \mathsf{sk}_{\mathsf{numKeys}}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ and given to $\mathcal{A}$. The current value of* $\mathsf{numKeys}$ *is added to* Cor *and* $\mathsf{numKeys}$ *is incremented.*
- Reencryption Key Generation $\mathcal{O}_{\mathsf{ReKeyGen}}$: *On input (i, j) where $i, j \le \mathsf{numKeys}$, if $i = j$ or if $i \in \mathsf{Hon}$ and $j \in \mathsf{Cor}$, output $\perp$. If $\mathcal{O}_{\mathsf{ReEnc}}$ has not been executed on input (i, j), compute and store $\mathsf{rk}_{i\to j} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$. Output the reencryption key $\mathsf{rk}_{i\to j}$*
- Reencryption $\mathcal{O}_{\mathsf{ReEnc}}$: *On input (i, j, ct_i) where $i, j \le \mathsf{numKeys}$, if $j \in \mathsf{Cor}$ and (i, ct_i) is a* derivative *of (i^*, ct^*), return $\perp$. Otherwise, let $\mathsf{ct}_j \leftarrow \mathsf{ReEnc}(\mathsf{rk}_{i\to j}, \mathsf{ct}_i)$, and return ct_j.*
- Decryption Oracle: *On input (i, ct) where $i \le \mathsf{numKeys}$, if the pair (i, ct) is a* derivative *of (i^*, ct^*), then return $\perp$. Otherwise, return $\mathsf{Dec}(\mathsf{sk}_i, \mathsf{ct})$.*
- Decision: *On input a bit $b' \in \{0,1\}$, return* win *if $b = b'$.*

The CCA advantage *of $\mathcal{A}$ is defined as*

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{cca}}(\lambda) = \Pr[\mathsf{win}],$$

where the probability is over the randomness of $\mathcal{A}$ and the oracles in CCA game.

[10] Unlike the CPA definition of [ABH09], the CCA definition in [CH07] does not divide the security game into three distinct phases. Rather, it allows calls Corrupted / Uncorrupted Key Generation calls to be made at any time.

Definition 11 (Derivative). Derivatives *of* (i^*, ct^*) *are defined inductively as follows.*

1. (i^*, ct^*) *is a derivative of itself.*
2. *If* $\mathcal{O}_{\mathsf{ReEnc}}$ *has been queried on input* (i, j, ct_i)*, returning output* ct_j*, then* (j, ct_j) *is a derivative of* (i, ct_i)*.*
3. *If* (i, ct) *is a derivative of* (i^*, ct^*)*, and* (i', ct') *is a derivative of* (i, ct)*, then* (i', ct') *is also a derivative of* (i^*, ct^*)*.*
4. *If* $\mathcal{O}_{\mathsf{ReKeyGen}}$ *has been queried on* (i, j) *and* $\mathsf{Dec}(j, \mathsf{ct}_j) \in \{\mathbf{m}_0, \mathbf{m}_1\}$*, then* (j, ct_j) *is a derivative of* (i, ct_i) *for all* ct_i*.*

The first three conditions prevent an adversary from learning the bit b' by a chain of reencryption queries resulting ending to a corrupted key or ending with a decryption query. The purpose of the fourth condition is the same: it applies the same protections to ciphertexts that the adversary reencrypts locally.

Definition 12 (Proxy Reencryption: CCA Security [ABH09]). *Given a security parameter* 1^λ*, a proxy reencryption scheme is CCA secure if for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, there exists a negligible function* negl *such that*

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{cca}}(\lambda) < \frac{1}{2} + \mathsf{negl}(\lambda)$$

B.2 CCA and HRA Security

We do not resolve the question of whether CCA security implies HRA security. It may seem that CCA security should imply HRA security. Intuitively, CCA security allows the adversary to make relatively unrestricted queries to both $\mathcal{O}_{\mathsf{ReEnc}}$ and $\mathcal{O}_{\mathsf{Dec}}$, whereas HRA restricts the adversary to making only honest reencryption queries to $\mathcal{O}_{\mathsf{ReEnc}}$. However the fourth type of derivative in the CCA definition restricts the CCA adversary in a way that stymies a naive attempt at a reduction.

The CCA definition of derivative is over-inclusive: it includes all ciphertexts that could in principle be derived from the challenge. On the other hand, the HRA security game restricts reencryption queries only when the ciphertext is actually a derivative of the challenge. The adversary may reencrypt other encryptions of the challenge messages, so long as those encryptions were honestly generated independently from the challenge ciphertext.

B.3 Separating IND-CCA$_{0,1}$ and IND-CCA$_{2,0}$

The definition of CCA security presented above grants the adversary access to $\mathcal{O}_{\mathsf{Dec}}$ and $\mathcal{O}_{\mathsf{ReEnc}}$ both before and after receiving the challenge ciphertext. Just as in the case of Enc-CCA-1 and Enc-CCA-2 security for standard encryption, it is natural to consider how the definition is altered by restricting the adversary's access to one or both oracles to the period before the challenge.

The work of [NAL15] formalize this problem by considering a family of security definitions $\mathsf{IND\text{-}CCA}_{d,r}$ parameterized by a pair of numbers $d, r \in \{0, 1, 2\}$. The parameter $d = 2$ means $\mathcal{O}_{\mathsf{Dec}}$ is *unrestricted*, $d = 1$ means that $\mathcal{O}_{\mathsf{Dec}}$ is *restricted* to before the challenge, and $d = 0$ means that $\mathcal{O}_{\mathsf{Dec}}$ is unavailable. Similarly, the parameter r defines the availability of $\mathcal{O}_{\mathsf{ReEnc}}$. $\mathsf{IND\text{-}CCA}_{2,2}$ corresponds to CCA security as defined above, whereas $\mathsf{IND\text{-}CCA}_{0,0}$ corresponds to CPA security.

In Theorems 4.6, [NAL15] show that $\mathsf{IND\text{-}CCA}_{2,0} \not\Longrightarrow \mathsf{IND\text{-}CCA}_{0,1}$. That is, even if a PRE scheme is secure with unrestricted access to $\mathcal{O}_{\mathsf{Dec}}$, it may be insecure with restricted access to $\mathcal{O}_{\mathsf{ReEnc}}$. We now prove a (stronger) converse. Our construction also demonstrates the failure of the intuition—described in the Introduction and motivating the original definition of CPA security—that access to $\mathcal{O}_{\mathsf{ReEnc}}$ is as powerful as $\mathcal{O}_{\mathsf{Dec}}$.

Theorem 6 ($\mathsf{IND\text{-}CCA}_{0,2} \not\Longrightarrow \mathsf{IND\text{-}CCA}_{1,0}$)**.** *If there exists a PRE scheme that is* $\mathsf{IND\text{-}CCA}_{0,2}$ *secure, then there exists a PRE scheme that is* $\mathsf{IND\text{-}CCA}_{0,2}$ *secure but not* $\mathsf{IND\text{-}CCA}_{1,0}$ *secure.*

Proof. Suppose PRE is $\mathsf{IND\text{-}CCA}_{0,2}$ secure, and let $\top$ be a special symbol that is not a valid ciphertext. Define a new scheme PRE' by modifying decryption as follows:

$$\mathsf{Dec}'(\mathsf{sk}, \mathsf{ct}) := \begin{cases} \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) & \text{if } \mathsf{ct} \neq \top \\ \mathsf{sk} & \text{if } \mathsf{ct} = \top \end{cases}.$$

PRE' is $\mathsf{IND\text{-}CCA}_{0,2}$ secure: without access to $\mathcal{O}_{\mathsf{Dec}}$, the view of the adversary is independent of whether the challenger uses PRE or PRE'.

PRE' is not $\mathsf{IND\text{-}CCA}_{1,0}$ secure: observe that a single call to $\mathcal{O}_{\mathsf{Dec}}(i^*, \top)$ allows the adversary to learn the challenge secret key sk_{i^*} and thereby distinguish encryptions of $\mathbf{m}_0$ and $\mathbf{m}_1$.

C Comparison to RIND-CPA

The concurrent work of Derler, Krenn, Lorünser, Ramacher, Slamanig, and Striecks identify the same problem with CPA security as discussed [DKL+18]. They define a new security notion—RIND-CPA security—as an additional property that proxy reencryptions schemes should guarantee.

The key feature of RIND-CPA security is that the adversary gets access to an unrestricted ReEnc oracle, but only before seeing the challenge ciphertext. The definition is similar to $\mathsf{IND\text{-}CCA}_{0,1}$ of [NAL15]. The definition of the RIND-CPA security experiment is from [DKL+18, Experiment 8].

Definition 13 (RIND-CPA Security Experiment).

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda), (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}), b \leftarrow \{0, 1\}$
$(\mathsf{pk}^*, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{pk})$
$\mathsf{rk} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}, \mathsf{pk}^*)$
$(\mathbf{m}_0, \mathbf{m}_1, \mathsf{st}) \leftarrow \mathcal{A}^{\{\mathsf{ReEnc}(\mathsf{rk}, \cdot)\}}(\mathsf{st})$

$b^* \leftarrow \mathcal{A}(\mathsf{st}, \mathsf{Enc}(\mathsf{pk}, \mathbf{m}_b))$
if $b = b^*$ *return 1, else return 0.*

RIND-CPA security requires that for all efficient adversaries, the probability of outputting 1 in the experiment is $\frac{1}{2} \pm \mathsf{negl}(\lambda)$.

In this section, we compare the approach of [DKL+18] with ours. We begin by describing RIND-CPA security as defined by [DKL+18]. Next, we compare RIND-CPA with HRA security informally, arguing that HRA provides the better generalization of Enc-CPA security to the PRE setting. Finally, we show that HRA and RIND-CPA security are incomparable notions.

C.1 Informal comparison

RIND-CPA is less suitable than HRA as a replacement for CPA security of proxy reencryption. First and most importantly, HRA better captures the intuitive guarantees of Enc-CPA security for standard encryption. Second, access to an unrestricted ReEnc oracle makes it a more useful as a testing ground for the development of new techniques. Finally, two idiosyncrasies of the [DKL+18] formulation of RIND-CPA security present additional issues.

Capturing Enc-CPA security. In Enc-CPA security for standard encryption, the adversary is able to arbitrarily affect the distribution of plaintext messages. One way of viewing this aspect of the definition is that Enc-CPA requires security while being agnostic as to the true distribution over messages (except that it is efficiently sampleable). Other than choosing the distribution over messages, the adversary is only allowed to see publicly-available information (i.e. public keys and parameters) and honestly encrypted ciphertexts. Informally, the Enc-CPA guarantee is that security should hold under normal operating conditions against eavesdropping parties without making distributional assumptions on plaintext messages. However, Enc-CPA makes no guarantees about dishonestly generated or malformed ciphertexts.

HRA security captures this intuitive guarantee better than RIND-CPA. In the course of normal operation of a proxy reencryption, an adversarial party will see reencrypted ciphertexts. These ciphertexts may come at any time—both before and after other ciphertexts whose contents should remain secret. While HRA allows reencryption both before and after the challenge, RIND-CPA restricts the reencryption oracle to the period before the challenge.

HRA makes minimal assumptions about the distribution of plaintext messages by allowing the adversary to choose messages itself, just as in Enc-CPA. RIND-CPA goes further by making requirements in the face of malformed or dishonestly generated ciphertexts.

A testing ground for new techniques. For classical encryption, Enc-CCA security is strictly stronger than Enc-CPA security. In fact, there are many settings where Enc-CPA security is demonstrably insufficient. Why then does the cryptography community continue to study it? There are many answers to this question, but

we mention only two. First, although insufficient for some applications, Enc-CPA is useful in others. Second, it is useful as an intermediate goal because it seems to capture a sort of hard core of the general problem of encryption and spurs the development of new techniques.

HRA security enjoys these same features; RIND-CPA does not. As for usefulness for applications, HRA is meaningful in many of the envisioned applications of proxy reencryption—many more than CPA security. Because RIND-CPA restricts the reencryption oracle to the period before the challenge ciphertext, its usefulness in applications is less clear.

The challenge of constructing CCA secure proxy reencryption is the same as the challenge of Enc-CCA secure encryption: namely, dealing with dishonestly generated, possibly malformed ciphertexts. RIND-CPA, by allowing malformed ciphertexts, presents similar challenges as full CCA security.

As for the usefulness of HRA as an intermediate goal towards CCA security, the historical development of proxy reencryption is proof itself. This sounds paradoxical: how can this be true if the notion has only just been introduced in this work? Many of cryptographers that were targeting CPA security developed schemes that achieve HRA security with only minimal modification. The techniques developed in these constructions were later adapted to achieve CCA security. This suggests that cryptographers' intuitions for the hard core of reencryption were not flawed, only the formalization of these intuitions as CPA security. HRA security is a better formalization for these intuitions and thus an appropriate intermediate goal for reencryption research.

Idiosyncrasies of the RIND-CPA definition. We mention two unusual properties of the [DKL+18] definition. Unlike the adversary's access to a ReEnc oracle, these properties are not inherent in the RIND-CPA concept. It would be easy to propose a modified RIND-CPA definition that did not have these properties (e.g., $\mathsf{IND\text{-}CCA}_{0,1}$ in [NAL15]).

First, the definition only considers the two party setting. Much like the informal description of proxy reencryption in Section 1, there is only a single uncorrupted key and a single corrupted key. It is easy to show that security in the two party setting does not imply security in a many party setting.

Second, not only are inputs to ReEnc allowed to be malformed, but the corrupted public key pk^* can be malformed as well. The adversary outputs pk^* itself and it needs not be honestly generated. This makes RIND-CPA security as defined in [DKL+18] formally incomparable to all other definitions of proxy reencryption security we know of, including the $\mathsf{IND\text{-}CCA}_{0,1}$ of [NAL15].

These drawbacks of the [DKL+18] definition do not affect the proof of Theorem 8, but neither does the proof depend on them.

C.2 Separating RIND-CPA and HRA security

The following pair of theorems support the conclusion that HRA security and RIND-CPA security are incomparable.

Theorem 7. *If there exists an HRA secure PRE scheme, then there exists a PRE scheme that is HRA secure but not RIND-CPA secure.*

Proof. Suppose PRE is HRA secure, and let $\top$ be a special symbol that is not a valid ciphertext. Define a new scheme PRE' by modifying reencryption as follows:

$$\mathsf{ReEnc}'(\mathsf{rk}, \mathsf{ct}) := \begin{cases} \mathsf{ReEnc}(\mathsf{rk}, \mathsf{ct}) & \text{if } \mathsf{ct} \neq \top \\ \mathsf{rk} & \text{if } \mathsf{ct} = \top \end{cases}.$$

PRE' is still HRA secure: $\mathcal{O}_{\mathsf{ReEnc}'}$ is functionally equivalent to $\mathcal{O}_{\mathsf{ReEnc}}$ when restricted to honestly generated ciphertexts..

PRE' is not RIND-CPA secure: a single call to $\mathcal{O}_{\mathsf{ReEnc}'}(i, j, \top)$ (made before the challenge) allows the adversary to learn the reencryption key $\mathsf{rk}_{i \to j}$ and thereby decrypt the challenge ciphertext.

Theorem 8. *Under the assumptions stated below, there exists a PRE scheme that is RIND-CPA secure but not HRA secure.*

The claim assumes the existence of pair of encryption schemes, PRE and FHE with the following properties. PRE is a RIND-CPA secure proxy reencryption scheme with a ciphertext space $\mathcal{C}_{\mathsf{inner}}$. FHE is a circuit private fully homomorphic encryption scheme with message space $\mathcal{M}_{\mathsf{fhe}} = \mathcal{C}_{\mathsf{inner}}$. The message spaces and ciphertext spaces of the two schemes are all disjoint and efficiently decidable. Finally, the additional proxy reencryption scheme $\mathsf{PRE}_{\mathsf{FHE}}$ corresponding to the FHE scheme (see Section 5.3) is RIND-CPA secure.[11] For simplicity, we also assume perfect correctness of reencryption (for both schemes) and of homomorphic evaluation.

The remainder of this appendix is devoted to the proof of Theorem 8.

Proof Intuition for Theorem 8 Recall that RIND-CPA security allows the adversary access to an unrestricted ReEnc oracle, but only before the challenge ciphertext is generated. The main difficulty in separating RIND-CPA and HRA security is the restriction in the HRA reencryption oracle to honestly generated ciphertexts.

The first idea in our construction is the observation that separating RIND-CPA and HRA security would be easy if it were possible to use Enc oracle to generate a fresh, honest encryption of the challenge plaintext. This fresh encryption could be reencrypted by the HRA reencryption oracle to a corrupted key, revealing the challenge plaintext.

The second idea is to have two layers of encryption, where the message space of the outer layer is equal to the ciphertext space of the inner layer. If the

[11] The proof requires that an encryption scheme be both fully homomorphic and support proxy reencryption with RIND-CPA security. For concreteness, we have chosen to assume that there exists an FHE scheme whose corresponding PRE is RIND-CPA secure, but a different construction would suffice. We do not further explore the underlying cryptographic assumptions needed to instantiate this encryption scheme.

challenge ciphertext comes from the inner layer, then it can be used as input to the Enc oracle to generate a new *outer ciphertext* containing information about the challenge plaintext—namely, an encryption of the challenge ciphertext. The outer ciphertext is honestly generated and can be reencrypted to a corrupt party and decrypted. But it seems we are no better off; decrypting the outer ciphertext only returns the challenge ciphertext still encrypted under the key of an honest party.

The third idea is to modify ReEnc—using fully homomorphic encryption—to reencrypt both the outer ciphertext and the inner challenge ciphertext. In addition to the usual reencrypted ciphertext, we augment ReEnc to output an additional, *doubly reencrypted* ciphertext, where both the outer and inner ciphertexts have been reencrypted. If the recipient of the resulting ciphertext is corrupt, the adversary can decrypt both layers and recover the challenge plaintext, violating HRA security.

We now describe the intuition for how to perform double reencryption. Suppose the proxy reencryption scheme used for the outer layer of encryption is also fully homomorphic. Such a scheme can be constructed from any FHE scheme (see Section 5.3). Given input an outer layer ciphertext $\mathsf{ct}_{\mathsf{outer}} = \mathsf{Enc}(\mathsf{ct}_{\mathsf{inner}})$, ReEnc will homomorphically evaluate $\mathsf{Eval}_{\mathsf{fhe}}(\mathsf{ReEnc}, \mathsf{ct}_{\mathsf{outer}})$. The result is an (non-reencrypted) outer ciphertext containing a reencrypted inner ciphertext. Then, ReEnc reencrypts that outer ciphertext. This produces a reencrypted outer ciphertext containing a reencrypted inner ciphertext.

Violating HRA security is simple: the adversary encrypts the challenge ciphertexts, reencrypts it to a corrupted key, then decrypts the doubly-reencrypted component twice to recover the challenge message.

It remains to prove that the constructed PRE scheme is RIND-CPA secure. Double reencryption can be simulated by a sequence of calls to Enc, Dec and $\mathcal{O}_{\mathsf{ReEnc}}$, allowing us to analyze the two-layered scheme without the double-reencryption modification to ReEnc. The RIND-CPA security of that scheme follows directly from the RIND-CPA security of the PRE scheme underlying the two layers.

The remainder of the section is organized as follows. First, we formalize *Dual PRE*, a conceptually simpler generalization of the layered encryption described above. We then present the PRE scheme used to separate RIND-CPA and HRA security. Finally, we show that the scheme is indeed HRA insecure and RIND-CPA secure.

Dual PRE Let $\mathsf{PRE}_\alpha = (\mathsf{Setup}_\alpha, \mathsf{KeyGen}_\alpha, \mathsf{Enc}_\alpha, \mathsf{Dec}_\alpha, \mathsf{ReKeyGen}_\alpha, \mathsf{ReEnc}_\alpha)$ be a proxy reencryption scheme with message space $\mathcal{M}_\alpha$ and ciphertext space $\mathcal{C}_\alpha$. Let $\mathsf{PRE}_\beta = (\mathsf{Setup}_\beta, \mathsf{KeyGen}_\beta, \mathsf{Enc}_\beta, \mathsf{Dec}_\beta, \mathsf{ReKeyGen}_\alpha, \mathsf{ReEnc}_\alpha)$ be a proxy reencryption scheme with message space $\mathcal{M}_\beta$ and ciphertext space $\mathcal{C}_\beta$. Suppose that $\mathcal{M}_\alpha$ and $\mathcal{M}_\beta$ are disjoint and that $\mathcal{C}_\alpha$ and $\mathcal{C}_\beta$ are disjoint. Suppose also that membership is efficiently decidable for all four sets. We define the *Dual PRE scheme* $\mathsf{PRE}_{\mathsf{dual}} = (\mathsf{Setup}_{\mathsf{dual}}, \mathsf{KeyGen}_{\mathsf{dual}}, \mathsf{Enc}_{\mathsf{dual}}, \mathsf{Dec}_{\mathsf{dual}}, \mathsf{ReKeyGen}_{\mathsf{dual}}, \mathsf{ReEnc}_{\mathsf{dual}})$ with mes-

sage space $\mathcal{M}_{\mathsf{dual}} = \mathcal{M}_\alpha \cup \mathcal{M}_\beta$ and ciphertext space $\mathcal{C}_{\mathsf{dual}} = \mathcal{C}_\alpha \cup \mathcal{C}_\beta$ as follows.

$$\begin{aligned}
\mathsf{Setup}_{\mathsf{dual}}(1^\lambda) &:= (\mathsf{pp}^\alpha, \mathsf{pp}^\beta), \\
&\quad \text{where } \mathsf{pp}^\alpha \leftarrow \mathsf{Setup}_\alpha(1^\lambda) \\
&\quad\quad \mathsf{pp}^\beta \leftarrow \mathsf{Setup}_\beta(1^\lambda) \\
\mathsf{KeyGen}_{\mathsf{dual}}(\mathsf{pp}^\alpha, \mathsf{pp}^\beta) &:= ((\mathsf{pk}^\alpha, \mathsf{pk}^\beta), (\mathsf{sk}^\alpha, \mathsf{sk}^\beta)), \\
&\quad \text{where } (\mathsf{pk}^\alpha, \mathsf{sk}^\alpha) \leftarrow \mathsf{KeyGen}_\alpha(\mathsf{pp}^\alpha) \\
&\quad\quad (\mathsf{pk}^\beta, \mathsf{sk}^\beta) \leftarrow \mathsf{KeyGen}_\beta(\mathsf{pp}^\beta)) \\
\mathsf{ReKeyGen}_{\mathsf{dual}}(\mathsf{sk}_i^\alpha, \mathsf{sk}_i^\beta, \mathsf{pk}_j^\alpha, \mathsf{pk}_j^\beta) &= (\mathsf{rk}_{i\to j}^\alpha, \mathsf{rk}_{i\to j}^\beta), \\
&\quad \text{where } \mathsf{rk}_{i\to j}^\alpha \leftarrow \mathsf{ReKeyGen}_\alpha(\mathsf{sk}_i^\alpha, \mathsf{pk}_j^\alpha) \\
&\quad\quad \mathsf{rk}_{i\to j}^\beta \leftarrow \mathsf{ReKeyGen}_\beta(\mathsf{sk}_i^\beta, \mathsf{pk}_j^\beta) \\
\mathsf{Enc}_{\mathsf{dual}}(\mathsf{pk}^\alpha, \mathsf{pk}^\beta, \mathbf{m}) &:= \begin{cases} \mathsf{Enc}_\alpha(\mathsf{pk}^\alpha, \mathbf{m}) \text{ if } \mathbf{m} \in \mathcal{M}_\alpha \\ \mathsf{Enc}_\beta(\mathsf{pk}^\beta, \mathbf{m}) \text{ if } \mathbf{m} \in \mathcal{M}_\beta \end{cases} \\
\mathsf{Dec}_{\mathsf{dual}}(\mathsf{sk}^\alpha, \mathsf{sk}^\beta, \mathsf{ct}) &:= \begin{cases} \mathsf{Dec}_\alpha(\mathsf{sk}^\alpha, \mathsf{ct}) \text{ if } \mathsf{ct} \in \mathcal{C}_\alpha \\ \mathsf{Dec}_\beta(\mathsf{sk}^\beta, \mathsf{ct}) \text{ if } \mathsf{ct} \in \mathcal{C}_\beta \end{cases} \\
\mathsf{ReEnc}_{\mathsf{dual}}(\mathsf{rk}^\alpha, \mathsf{rk}^\beta, \mathsf{ct}) &:= \begin{cases} \mathsf{ReEnc}_\alpha(\mathsf{rk}^\alpha, \mathsf{ct}) \text{ if } \mathsf{ct} \in \mathcal{C}_\alpha \\ \mathsf{ReEnc}_\beta(\mathsf{rk}^\beta, \mathsf{ct}) \text{ if } \mathsf{ct} \in \mathcal{C}_\beta \end{cases}
\end{aligned}$$

Claim. If PRE_α and PRE_β are RIND-CPA secure, then $\mathsf{PRE}_{\mathsf{dual}}$ is RIND-CPA secure.

Proof. For any efficient adversary $\mathcal{A}_{\mathsf{dual}}$, there exist efficient adversaries $\mathcal{A}_\alpha$ and $\mathcal{A}_\beta$ such that $\mathsf{Adv}^{\mathcal{A}_{\mathsf{dual}}}_{\mathsf{PRE}_{\mathsf{dual}}} \leq \max\{\mathsf{Adv}^{\mathcal{A}_\alpha}_{\mathsf{PRE}_\alpha}, \mathsf{Adv}^{\mathcal{A}_\beta}_{\mathsf{PRE}_\beta}\}$. The adversary $\mathcal{A}_\alpha$ (respectively, $\mathcal{A}_\beta$) simulates the $\mathsf{PRE}_{\mathsf{dual}}$ security game by performing either calling its own oracles or simulating an instance of PRE_β (resp., $\mathcal{A}_\alpha$), as appropriate. By the assumption that PRE_α and PRE_β are both RIND-CPA secure, $\mathsf{Adv}^{\mathcal{A}_{\mathsf{dual}}}_{\mathsf{PRE}_{\mathsf{dual}}}$ is negligible.

The Counterexample For the relevant background on fully homomorphic encryption and its connection to proxy reencryption, see Section 5.3 and [Gen09, Section 2].

Let $\mathsf{PRE}_{\mathsf{inner}}$ be a RIND-CPA secure proxy reencryption scheme with ciphertext space $\mathcal{C}_{\mathsf{inner}}$. Let FHE be a circuit private fully homomorphic encryption scheme with message space $\mathcal{M}_{\mathsf{fhe}} = \mathcal{C}_{\mathsf{inner}}$ and let $\mathsf{PRE}_{\mathsf{fhe}}$ be the corresponding proxy reencryption scheme. Suppose that $\mathsf{PRE}_{\mathsf{fhe}}$ is RIND-CPA secure. Note that FHE and $\mathsf{PRE}_{\mathsf{fhe}}$ are two different views of *the same encryption scheme*, with identical KeyGen, Enc, and Dec algorithms. Specifically, the homomorphic evaluation algorithm $\mathsf{Eval}_{\mathsf{fhe}}$ works on ciphertexts of $\mathsf{PRE}_{\mathsf{fhe}}$.

We construct a proxy reencryption scheme PRE by modifying the Dual PRE scheme. Let $\mathsf{PRE}_{\mathsf{dual}}$ be the Dual PRE of $\mathsf{PRE}_\alpha = \mathsf{PRE}_{\mathsf{inner}}$ and $\mathsf{PRE}_\beta = \mathsf{PRE}_{\mathsf{fhe}}$. We augment $\mathsf{ReEnc}_{\mathsf{dual}}$ to perform double reencryption as described in the proof

intuition above in addition to the normal ciphertext reencryption. Enc and Dec are modified accordingly. The modified lines are marked with a $\star$.

$$\mathsf{Enc}(\mathsf{pk}^{\mathsf{inner}}, \mathsf{pk}^{\mathsf{fhe}}, \mathbf{m}) := \begin{cases} \mathsf{Enc}_{\mathsf{inner}}(\mathsf{pk}^{\mathsf{inner}}, \mathbf{m}) & \text{if } \mathbf{m} \in \mathcal{M}_{\mathsf{inner}} \\ \big(\mathsf{Enc}_{\mathsf{fhe}}(\mathsf{pk}^{\mathsf{fhe}}, \mathbf{m}), \bot\big) & \text{if } \mathbf{m} \in \mathcal{M}_{\mathsf{fhe}} \quad (\star) \end{cases}$$

$$\mathsf{ReEnc}(\mathsf{rk}^{\mathsf{inner}}, \mathsf{rk}^{\mathsf{fhe}}, \mathsf{ct}) := \begin{cases} \mathsf{ReEnc}_{\mathsf{inner}}(\mathsf{rk}^{\mathsf{inner}}, \mathsf{ct}) & \text{if } \mathsf{ct} \in \mathcal{C}_{\mathsf{inner}} \\ \big(\mathsf{ReEnc}_{\mathsf{fhe}}(\mathsf{rk}^{\mathsf{fhe}}, \mathsf{ct}^1), \widetilde{\mathsf{ct}}\big) & \text{if } \mathsf{ct} = (\mathsf{ct}^1, \mathsf{ct}^2) \text{ and } \mathsf{ct}^1 \in \mathcal{C}_{\mathsf{fhe}} \quad (\star) \\ \bot & \text{otherwise} \quad (\star) \end{cases}$$

$$\text{where } \widetilde{\mathsf{ct}} = \mathsf{ReEnc}_{\mathsf{fhe}}\Big(\mathsf{rk}^{\mathsf{fhe}}, \mathsf{Eval}_{\mathsf{fhe}}\big(\mathsf{ReEnc}_{\mathsf{inner}}(\mathsf{rk}^{\mathsf{inner}}, \cdot), \mathsf{ct}^1\big)\Big)$$

$$\mathsf{Dec}(\mathsf{sk}^{\mathsf{inner}}, \mathsf{sk}^{\mathsf{fhe}}, \mathsf{ct}) := \begin{cases} \mathsf{Dec}_{\mathsf{inner}}(\mathsf{sk}^{\mathsf{inner}}, \mathsf{ct}) & \text{if } \mathsf{ct} \in \mathcal{C}_{\mathsf{inner}} \\ \mathsf{Dec}_{\mathsf{fhe}}(\mathsf{sk}^{\mathsf{fhe}}, \mathsf{ct}^1) & \text{if } \mathsf{ct} = (\mathsf{ct}^1, \mathsf{ct}^2) \text{ and } \mathsf{ct}^1 \in \mathcal{C}_{\mathsf{fhe}} \quad (\star) \end{cases}$$

Observe that for honestly generated ciphertexts, $\mathsf{ct} = (\mathsf{ct}^1, \mathsf{ct}^2)$ if and only if $\mathsf{ct}^1 \in \mathcal{C}_{\mathsf{fhe}}$, and both hold whenever $\mathsf{ct} \notin \mathcal{C}_{\mathsf{inner}}$.

HRA Insecurity The attack on the HRA security game uses only the correctness of the PRE and FHE schemes. It requires only two parties; to help disambiguate the various subscripts and superscripts, we will index these parties by alice and bob. Let $\mathbf{m}_0, \mathbf{m}_1 \in \mathcal{M}_{\mathsf{inner}}$ be distinct messages. The HRA adversary proceeds as follows.

- Receive pp from challenger. Call the Uncorrupted Key Generation oracle once to obtain $\mathsf{pk}_{\mathsf{alice}}$. Call the Corrupted Key Generation oracle once to obtain $(\mathsf{pk}_{\mathsf{bob}}, \mathsf{sk}_{\mathsf{bob}})$.
- Query the Challenge oracle on input $(\mathsf{alice}, \mathbf{m}_0, \mathbf{m}_1)$ to obtain the challenge $(1, \mathsf{ct}^*_{\mathsf{alice},\mathsf{inner}})$, an encryption of $\mathbf{m}_b$ for random $b \leftarrow \{0,1\}$. Recall that 1 is the index of this ciphertext which may be used as input to $\mathcal{O}_{\mathsf{ReEnc}}$.
- Query $\mathcal{O}_{\mathsf{Enc}}(\mathsf{alice}, \mathsf{ct}^*_{\mathsf{alice},\mathsf{inner}})$ to obtain the ciphertext $(2, \mathsf{ct}_{\mathsf{alice}})$, where $\mathsf{ct}_{\mathsf{alice}} = (\mathsf{ct}^*_{\mathsf{alice},\mathsf{fhe}}, \bot)$.
- Query $\mathcal{O}_{\mathsf{ReEnc}}(\mathsf{alice}, \mathsf{bob}, 2)$ to obtain $(3, \mathsf{ct}_{\mathsf{bob}})$, where $\mathsf{ct}_{\mathsf{bob}} = (\mathsf{ct}^1_{\mathsf{bob},\mathsf{fhe}}, \mathsf{ct}^*_{\mathsf{bob},\mathsf{fhe}})$. If $\mathcal{O}_{\mathsf{ReEnc}}$ outputs $\bot$, abort.
- Let $\mathsf{ct}^*_{\mathsf{bob},\mathsf{inner}} = \mathsf{Dec}\big(\mathsf{sk}_{\mathsf{bob}}, (\mathsf{ct}^*_{\mathsf{bob},\mathsf{fhe}}, \bot)\big)$. Let $\mathbf{m}' = \mathsf{Dec}(\mathsf{sk}_{\mathsf{bob}}, \mathsf{ct}^*_{\mathsf{bob},\mathsf{inner}})$.
- Finally, call the Decision oracle with b' such that $\mathbf{m}_{b'} = \mathbf{m}'$.

The call to $\mathcal{O}_{\mathsf{ReEnc}}$ never causes the adversary to abort, because no ciphertext output by $\mathcal{O}_{\mathsf{Enc}}$ can be in the set Deriv (see the specification of $\mathcal{O}_{\mathsf{ReEnc}}$ in Definition 5).

It remains to show only that $b' = b$. By the correctness of reencryption and homomorphic evaluation for $\mathsf{PRE}_{\mathsf{fhe}}$:

$$\begin{aligned} \mathsf{ct}^*_{\mathsf{bob},\mathsf{inner}} &= \mathsf{Dec}_{\mathsf{fhe}}(\mathsf{sk}^{\mathsf{fhe}}_{\mathsf{bob}}, \mathsf{ct}^*_{\mathsf{bob},\mathsf{fhe}}) \\ &= \mathsf{Dec}_{\mathsf{fhe}}\left(\mathsf{sk}^{\mathsf{fhe}}_{\mathsf{bob}}, \mathsf{ReEnc}_{\mathsf{fhe}}\left(\mathsf{rk}^{\mathsf{fhe}}_{\mathsf{alice}\to\mathsf{bob}}, \mathsf{Eval}_{\mathsf{fhe}}\big(\mathsf{ReEnc}_{\mathsf{inner}}(\mathsf{rk}^{\mathsf{inner}}_{\mathsf{alice}\to\mathsf{bob}}, \cdot), \mathsf{ct}^*_{\mathsf{alice},\mathsf{fhe}}\big)\right)\right) \end{aligned}$$

$$= \mathsf{ReEnc}_{\mathsf{inner}}\left(\mathsf{rk}^{\mathsf{inner}}_{\mathsf{alice}\to\mathsf{bob}}, \mathsf{ct}^{*}_{\mathsf{alice,inner}}\right)$$

By the correctness of $\mathsf{PRE}_{\mathsf{inner}}$:

$$\begin{aligned}
\mathbf{m}_{b'} &= \mathsf{Dec}_{\mathsf{inner}}(\mathsf{sk}^{\mathsf{inner}}_{\mathsf{bob}}, \mathsf{ct}^{*}_{\mathsf{bob,inner}}) \\
&= \mathsf{Dec}_{\mathsf{inner}}\left(\mathsf{sk}^{\mathsf{inner}}_{\mathsf{bob}}, \mathsf{ReEnc}_{\mathsf{inner}}\left(\mathsf{rk}^{\mathsf{inner}}_{\mathsf{alice}\to\mathsf{bob}}, \mathsf{ct}^{*}_{\mathsf{alice,inner}}\right)\right) \\
&= \mathsf{Dec}_{\mathsf{inner}}\left(\mathsf{sk}^{\mathsf{inner}}_{\mathsf{bob}}, \mathsf{ReEnc}_{\mathsf{inner}}\left(\mathsf{rk}^{\mathsf{inner}}_{\mathsf{alice}\to\mathsf{bob}}, \mathsf{Enc}_{\mathsf{inner}}\left(\mathsf{pk}^{\mathsf{inner}}_{\mathsf{alice}}, \mathbf{m}_b\right)\right)\right) \\
&= \mathbf{m}_b
\end{aligned}$$

RIND-CPA Security We assumed that both $\mathsf{PRE}_{\mathsf{inner}}$ and $\mathsf{PRE}_{\mathsf{fhe}}$ were RIND-CPA secure. By Claim C.2, the Dual PRE scheme $\mathsf{PRE}_{\mathsf{dual}}$ is also RIND-CPA secure. It remains to show that the counterexample PRE (constructed by modifying $\mathsf{PRE}_{\mathsf{dual}}$) is RIND-CPA secure.

We reduce the RIND-CPA security of PRE to the RIND-CPA security of $\mathsf{PRE}_{\mathsf{dual}}$. Recall that the RIND-CPA security game is like the CPA security game, but with access to an unrestricted reencryption oracle $\mathcal{O}_{\mathsf{ReEnc}}$ up until the Challenge Oracle is called.

From any probabilistic, polynomial-time algorithm $\mathcal{A}$ (the PRE adversary), we construct an algorithm $\mathcal{A}' = \mathcal{A}_{\mathsf{dual}}$ such that $\mathsf{Adv}^{\mathcal{A}'}_{\mathsf{rind-cpa}}(\lambda) \geq \mathsf{Adv}^{\mathcal{A}}_{\mathsf{rind-cpa}}(\lambda) - \mathsf{negl}(\lambda)$; by the RIND-CPA security of $\mathsf{PRE}_{\mathsf{dual}}$ this advantage is negligible, completing the proof.

$\mathcal{A}_{\mathsf{dual}}$ runs $\mathcal{A}$ and simulates the RIND-CPA security game (if $\mathcal{A}$ does not follow the specification of the HRA security game, $\mathcal{A}_{\mathsf{dual}}$ simply aborts). At the outset, $\mathcal{A}_{\mathsf{dual}}$ makes one additional Corrupted Key Generation query, obtaining the key pair $(\mathsf{pk}_\star, \mathsf{sk}_\star)$. This key pair is not passed to $\mathcal{A}$, and will only be used to answer calls to $\mathcal{O}_{\mathsf{ReEnc}}$. To answer oracle calls by $\mathcal{A}$ to any oracle other than $\mathcal{O}_{\mathsf{ReEnc}}$ and $\mathcal{O}_{\mathsf{Enc}}$, $\mathcal{A}_{\mathsf{dual}}$ simply passes the calls and answers unaltered to the corresponding RIND-CPA oracles. These oracles are perfectly simulated.

To answer oracle calls to $\mathcal{O}_{\mathsf{Enc}}$: $\mathcal{A}_{\mathsf{dual}}$ calls its own encryption oracle to obtain the response (k, ct). It then returns $(k, (\mathsf{ct}, \bot))$ to $\mathcal{A}$. This perfectly simulates the encryption oracle for PRE.

To answer oracle calls to $\mathcal{O}_{\mathsf{ReEnc}}(i, j, \mathsf{ct})$: $\mathcal{A}_{\mathsf{dual}}$ uses a series of calls to its own reencryption oracle to construct the necessary ciphertext, as follows. For intuition, recall that the unrestricted reencryption oracle allows $\mathcal{A}_{\mathsf{dual}}$ to reencrypt any outer layer ciphertext to corrupted party $\star$ and thereby decrypt and learn the inner layer ciphertext. Then it can reencrypt the inner ciphertext, encrypt the result using the outer scheme using pk_i, and finally reencrypt the result to the recipient party j.

- If $\mathsf{ct} \in \mathcal{C}_{\mathsf{inner}}$, call own $\mathcal{O}_{\mathsf{ReEnc}}(i, j, \mathsf{ct})$, and return response to $\mathcal{A}$. If $\mathsf{ct} \neq (\mathsf{ct}^1, \mathsf{ct}^2)$ or $\mathsf{ct}^1 \notin \mathcal{C}_{\mathsf{fhe}}$, return $\bot$ to $\mathcal{A}$.
- Let $\mathsf{ct} = (\mathsf{ct}^1, \mathsf{ct}^2)$. Call own $\mathcal{O}_{\mathsf{ReEnc}}(i, j, \mathsf{ct}^1)$ and let ct^1_j be the response.
- Call own $\mathcal{O}_{\mathsf{ReEnc}}(i, \star, \mathsf{ct}^1)$ and let $\mathsf{ct}^1_\star$ be the response. Note that $\mathsf{ct}^1_\star$ is the encryption under $\mathsf{pk}_\star$ of a *ciphertext.* Let $\mathsf{ct}' = \mathsf{Dec}(\mathsf{sk}_\star, \mathsf{ct}^1_\star)$.

– Call own $\mathcal{O}_{\mathsf{ReEnc}}(i, j, \mathsf{ct}')$, and let ct'_j be the response.
– Call own $\mathcal{O}_{\mathsf{ReEnc}}(i, j, \mathsf{Enc}(\mathsf{pk}_i, \mathsf{ct}'_j))$, and let $\widetilde{\mathsf{ct}}$ be the response.
– Return $(\mathsf{ct}^1_j, \widetilde{\mathsf{ct}})$ to $\mathcal{A}$.

This whole sequence of reencryptions, decryptions, evaluations, and encryptions produces a ciphertext whose output distribution is statistically close to the output distribution of $\mathcal{O}_{\mathsf{ReEnc}}$ in the real PRE game. This follows from the circuit privacy of the FHE scheme (along with the correctness of both the reencryption and FHE evaluation).

The views of $\mathcal{A}$ in the real game (using the real $\mathcal{O}_{\mathsf{ReEnc}}$) and the simulated security game (using the simulation described above) are statistically close. $\mathcal{A}_{\mathsf{dual}}$ wins the security game for $\mathsf{PRE}_{\mathsf{dual}}$ if and only if $\mathcal{A}$ wins the simulated security game for PRE.