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

Aloni Cohen[*]

MIT

Abstract

Consider three parties: Alice, Bob, and Polly. Alice keeps some encrypted data that she can decrypt with a secret key known to her. She wants to communicate the data to Bob, but not to Polly (nor anybody else). Assuming Alice knows Bob's public key, how can she communicate the data to him? Proxy reencryption provides an elegant answer: Alice creates a *reencryption key* that will enable Polly (the *proxy*) to reencrypt her data for Bob's use, but that 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 security against both Polly and Bob.

However, we observe 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. In common applications of proxy reencryption, this means that CPA security provides scant guarantees.

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

1 Introduction

Consider three parties: Alice, Bob, and Polly. Alice keeps some encrypted data that she can decrypt with a secret key known to her. She wants to communicate the data to Bob, but not to Polly (nor anybody else). Assuming Alice knows 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 may simply 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 answer: Alice creates a *reencryption key* that will enable Polly (the *proxy*) to reencrypt her data for Bob's use, but that will not reveal the data

[*]`aloni@mit.edu`. Supported in part by NSF GRFP, NSF MACS CNS-1413920, DARPA IBM W911NF-15-C-0236, and Simons Investigator Award Agreement Dated 6-5-12

to Polly. Alice wants Bob to recover the data uncorrupted (correctness) and wants to be sure that Polly cannot learn anything about her data (security).

But what about Bob? As already observed, if we do not require any security against Bob, proxy reencryption is trivial: Alice simply sends Bob her secret key (encrypted with his public key). Of course, this is undesirable, unsatisfying, and insufficient for a number of supposed applications of proxy reencryption (see Section 2).

Surprisingly, **the Trivial Scheme is a CPA-secure proxy reencryption** (defined below) when instantiated with a circular-secure encryption scheme [BHHO08]! The circular security is used only to prove security against a malicious Polly; Bob completely learns Alice's secret key. Relatedly, 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 any message encrypted under Alice's public key.[1]

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, a lack of security against Bob is inherent.

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, ABPW13, PWA$^+$16, FL], implementations [LPK10, HHY11, PRSV17, BPRR17], and early success in program obfuscation [HRsV11].

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, HRsV11], especially in latticed-based constructions [Gen09, ABPW13, PWA$^+$16, PRSV17].

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; they differ in the information granted to the adversary. CCA security [CH07] allows the adversary to corrupt either Bob (learning $\mathsf{sk}_{\mathsf{bob}}$) or Polly (learning the reencryption key rk), and additionally 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 the reencrypted ciphertext $\mathsf{ct}_{\mathsf{bob}}$.

This naturally extends the chosen-ciphertext security notion of standard encryption (Enc-CCA).

CPA security of proxy reencryption [AFGH06] also allows the adversary to corrupt either Bob or Polly, but removes both oracles. To adapt the chosen-plaintext security notion from standard encryption (Enc-CPA) to proxy reencryption, we must of course do away with $\mathcal{O}_{\mathsf{Dec}}$. It seems we must also remove $\mathcal{O}_{\mathsf{ReEnc}}$, sinceby corrupting Bob (learning $\mathsf{sk}_{\mathsf{bob}}$), the adversary can simulate $\mathcal{O}_{\mathsf{Dec}}$ by reencrypting and decrypting [ABH09]. Therefore removing both oracles naturally extends the Enc-CPA security notion to proxy reencryption.

[1]This can be formalized with some care. See also Footnote 2.

Unfortunately, a natural definition is not always a good definition. Not only is the above intuition false,[2] but CPA security as defined above guarantees little against a corrupted Bob: 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, allowing us to prove that the Trivial Scheme is CPA secure. Furthermore, CPA security is ill-suited for the most commonly cited applications of proxy reencryption, including forwarding of encrypted email and single-writer, many-reader encrypted storage (See Section 2).

What guarantee do we want from (unidirectional) proxy reencryption? For this overview, let's consider restrict ourselves to the three-party setting of Alice, Bob, and Polly described above. First, we want security against the 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 $\mathsf{pk}_{\mathsf{bob}}$. Security against the proxy is guaranteed by CPA.

Second, we want security against the receiver Bob (symmetrically, Alice) 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 have seen, security against the receiver is not guaranteed by CPA.[3]

Generalizing these dual guarantees to many possibly colluding parties, we want security as long as the adversary only sees honestly reencrypted ciphertexts. In Section 3, we formalize this notion as proxy reencryption security against *honest-reencryption attacks (PRE-HRA)*.

As already observed, CPA security for proxy reencryptionis a natural generalization of Enc-CPA security for (standard) encryption. Observe that Enc-CPA security does not change when the adversary is given access to an encryption oracle and an oracle that decrypts *honest* ciphertexts: those output by the encryption oracle (excluding the challenge ciphertext). PRE-HRA can be viewed as an adaptation of this equivalent view of Enc-CPA to proxy reencryption.

In addition to better capturing our intuitions about proxy reencryption security, PRE-HRA guarantees more meaningful security in the most common applications of proxy reencryption. PRE-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.

While PRE-HRA security is (strictly) stronger than CPA, it is (strictly) weaker than CCA. With the new definition, the most immediate question is: can we construct a proxy reencryption scheme that is PRE-HRA secure (that is not also CCA secure)?

A natural place to begin is with existing CPA schemes. In order to avoid proving the PRE-HRA

[2] It is easy to separate the notions. Suppose $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{ReKeyGen}, \mathsf{ReEnc})$ is CCA secure, and let "$\|$" denote concatenation of strings. Define a new scheme as follows: $\mathsf{KeyGen}' \equiv \mathsf{KeyGen}$; $\mathsf{ReKeyGen}' \equiv \mathsf{ReKeyGen}$; $\mathsf{Enc}'(\mathsf{pk}, \mathbf{m}) := \mathsf{Enc}(\mathsf{pk}, \mathbf{m})\|0$; $\mathsf{ReEnc}'(\mathsf{rk}, \mathsf{ct}\|b) := \mathsf{ReEnc}(\mathsf{rk}, \mathsf{ct})\|b$;

$$\mathsf{Dec}'(\mathsf{sk}, \mathsf{ct}\|b) := \begin{cases} \mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) & b = 0 \\ \mathsf{sk} & b = 1 \end{cases} .$$

This scheme enjoys security against an adversary with access to $\mathcal{O}_{\mathsf{ReEnc}}$, but is easily attacked using $\mathcal{O}_{\mathsf{Dec}}$.

[3] This dual-guarantee conception of proxy reencryption security mirrors the security requirements of [ID03] that were not adopted by subsequent work. Indeed, our definition of security can be viewed as an adaptation and modernization of their CPA notion (defined only in a proof in the appendix).

security of these schemes from first principles, we identify a property – *reencryption simulatablity* – which is sufficient to boost CPA security to PRE-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 an algorithm with access to the CPA oracles to efficiently implement the honest reencryption oracle, thereby reducing PRE-HRA security to CPA security.

In Section 4, we first examine the simple construction of proxy reencryption from any fully-homomorphic encryption [Gen09], and second the pairings-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.[4]

2 Insufficiency of CPA security for applications

In Section 3.1, we define CPA security of proxy reencryption, and formalize the Trivial Scheme from the introduction satisfying the notion. 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 in which CPA security (as implemented by the Trivial Scheme) is potentially unsatisfactory.[5] We revisit these applications in Section 3.3 after proposing a 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 is does not offer meaningful security against Bob if Alice only wishes to forward a subset of emails (for example, from particular senders or those received while on vacation).

Key escrow [ID03] Similar to email forwarding, "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" [ID03]. As in email forwarding, the "for a limited period of time"

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

[5]One might look to the originators of the proxy encryption notion: "Clearly, A must (unconditionally) trust B, since the encryption proxy function by definition allows B to decrypt on behalf of A" [BBS98]. While seeming to disagree with us, to properly understand [BBS98], it is important to recognize that the authors conceive of only two parties (A tells B the reencryption key). The shortcoming we identify does not manifest in that setting and therefore [BBS98] provides little guidance. We might also appeal 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.

requirement suggests that Ivan and Dodis would not have been satisfied with the Trivial Scheme.[6]

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 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 two applications of proxy reencryption for which CPA security does suffice.

Key rotation for encrypted cloud storage Encryption is a natural option when outsourcing storage to an untrusted cloud. Periodically updating secret keys is recommended by NIST, the Payment Card Industry Data Security Standard, and the OpenWeb Application Security Project [PWA+16]. Proxy reencryption naturally allows the storage server to perform the key rotation without decrypting. In this application, CPA security suffices as there are only two parties: the client (who is both Alice and Bob) and the server (Polly).

Fully homomorphic encryption [Gen09] Though not exactly an application of proxy reencryption, fully homomorphic encryption is closely connected. There is a trivial construction of proxy reencryption (unidirectional, multi-hop) from any public key fully homomorphic encryption.[7] Conversely, reencryption is a critical component of FHE constructions including Gentry's. As in key rotation, there is only one party with knowledge of secret keys; a homomorphic evaluator acts only as the proxy.

3 Security against honest reencryption attacks

In this section, we motivate and formalize security of proxy reencryption (PRE) against *honest reencryption attacks (PRE-HRA)*. 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). For the sake of concreteness, simplicity, and brevity, we restrict the discussion to unidirectional, single-hop schemes. In multi-hop schemes, $\mathsf{rk}_{A\to B}$ 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

[6]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.

[7]Though this fact was observed in [Gen09] and discussed in [Kir], it goes surprisingly unmentioned in a number of works constructing proxy reencryption from lattice assumptions [FL, PRSV17, PWA+16]. The construction also works from depth-bounded FHE (if the depth-bound is greater than the depth of the decryption circuit), removing the need for circularity assumptions.

or multi-hop schemes each have their benefits and drawbacks, and works typically focus on one or the other notion.[8] To the best of our knowledge, our observations and results can all be adapted to the multi-hop setting.

3.1 Security against chosen plaintext attacks

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

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$. *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.*
- $\mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j) \to \mathsf{rk}_{i\to j}$. *Given 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 algorithms *outputs a ciphertext* ct_i.
- $\mathsf{ReEnc}(\mathsf{rk}_{i\to j}, \mathsf{ct}_i) \to \mathsf{ct}_j$. *Given a reencryption key from* i *to* j *and a ciphertext under key* pk_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* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$ *and all* $\mathbf{m} \in \mathcal{M}$*:*

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

- *For all* $(\mathsf{pk}_i, \mathsf{sk}_i), (\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$*,* $\mathsf{rk}_{i\to j} \leftarrow \mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j)$*, and* $\mathbf{m} \in \mathcal{M}$*:*

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

[8] The literature is divided about whether "single-hop" is merely a correctness property (i.e., demonstrating how to reencrypt 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, HRsV11, NAL15], the oracle $\mathcal{O}_{\mathsf{ReKeyGen}}$ in the security game will not accept queries (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, FL], the oracle will answer such queries, but the challenge ciphertext must be encrypted under a key i^* for which no such reencryption key was generated (which can be formalized in a number of ways).

Security is modeled by a game played by an adversary $\mathcal{A}$. In the setting of many users, $\mathcal{A}$ 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* $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$*. $\mathcal{A}$ is given* pk_i*. The current value of* $\mathsf{numKeys}$ *is added to* Hon *and* $\mathsf{numKeys}$ *is incremented.*
- Corrupted Key Generation: *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.*

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*

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

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*

$$\mathsf{Adv}^{\mathcal{A}}_{CPA}(\lambda) < \mathsf{negl}(\lambda)$$

Remark 1. *Another definitional subtlety of proxy reencryption worth discussing affects not just CPA security, but PRE-HRA and CCA security as well. Consider the specification of $\mathcal{O}_{\mathsf{ReKeyGen}}$ and $\mathcal{O}_{\mathsf{ReEnc}}$ in Definition 3 of CPA security. 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 [ABPW13, FL], 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; it is a simple exercise to construct proxy reencryption schemes separating the notions. 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. Even so, we believe that it is generally desirable to use 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.

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. While it is easy to show that the following schemes (described in the introduction) are CPA secure, in each scheme a single ciphertext reencrypted from an honest index to a corrupted index completely destroys security.

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

$$\mathsf{ReEnc}'(\mathsf{rk}, \mathbf{m}) := \mathsf{ReEnc}(\mathsf{rk}, \mathbf{m}) \| \mathsf{rk}.$$

Trivial Scheme Let $(\mathsf{Setup}_{\mathsf{circ}}, \mathsf{KeyGen}_{\mathsf{circ}}, \mathsf{Enc}_{\mathsf{circ}}, \mathsf{Dec}_{\mathsf{circ}})$ be a circularly secure encryption scheme (or rather "clique-secure" as in [BHHO08]). Let $\mathsf{Setup} \equiv \mathsf{Setup}_{\mathsf{circ}}$; $\mathsf{KeyGen} \equiv \mathsf{KeyGen}_{\mathsf{circ}}$; $\mathsf{Enc} \equiv \mathsf{Enc}_{\mathsf{circ}}$; $\mathsf{ReKeyGen}(\mathsf{sk}_i, \mathsf{pk}_j) := \mathsf{Enc}(\mathsf{pk}_j, \mathsf{sk}_i)$; $\mathsf{ReEnc}(\mathsf{rk}_{i\to j}, \mathsf{ct}_i) := \mathsf{ct}_i \| \mathsf{rk}$;

$$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}) := \begin{cases} \mathsf{Dec}_{\mathsf{circ}}(\mathsf{Dec}_{\mathsf{circ}}(\mathsf{sk}, \mathsf{ct}''), \mathsf{ct}') & \text{if } \mathsf{ct} = \mathsf{ct}' \| \mathsf{ct}'' \\ \mathsf{Dec}_{\mathsf{circ}}(\mathsf{sk}, \mathsf{ct}) & \text{else} \end{cases}.$$

3.2 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 (PRE-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 definition 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. Following [CH07], our definition rules out this 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.

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

Phase 1:

- Setup: *Setup additionally initializes a counter* numCt *to 0, a key-value store $\mathcal{C}$ to empty, and a set* Deriv *to be empty.*

Phase 2:

- 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 $(\mathsf{numCt}, \mathsf{ct})$.*

- 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* Deriv*. Store the value* ct^* *in $\mathcal{C}$ with key (i, numCt). Return $(\mathsf{numCt}, \mathsf{ct}^*)$. This oracle can only be queried once.*

- 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). Return $(\mathsf{numCt}, \mathsf{ct}_j)$.*

Phase 3:

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

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

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

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

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

$$\mathsf{Adv}^{\mathcal{A}}_{PRE-HRA}(\lambda) < \mathsf{negl}(\lambda)$$

3.3 Sufficiency of PRE-HRA security for applications

Neither the Trivial Scheme nor the Concatenation Scheme satisfy PRE-HRA. Returning to the applications of proxy reencryption described in Section 2, we observe that PRE-HRA security provides meaningfully stronger security guarantees.

Encrypted email forwarding Using proxy reencryption with PRE-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 PRE-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), PRE-HRA security can support fine grained access control (i.e., each reader may access only a specific subset of the data).

4 Security of existing proxy reencryption schemes

Can we construct PRE-HRA-secure proxy reencryption? The most natural place to begin is with existing schemes. In this section we examine the relationships between CPA, PRE-HRA, and CCA security, showing that each is stronger than the previous. Although CPA security is strictly weaker than PRE-HRA security, we might hope that the existing CPA secure schemes already satisfy the more stringent definition. To this end, we identify a property (reencryption simulatability) sufficient to boost CPA security to PRE-HRA security.[9]

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 PRE-HRA secure in general, if the FHE is *circuit private* – a property Gentry imbues into his FHE by rerandomization – the same construction will be PRE-HRA secure. We then examine pairings-based schemes, finding there too that rerandomization provides a direct path to PRE-HRA security.[10]

4.1 PRE-HRA in relation to CPA and CCA security

In this section, we examine the relationships between PRE-HRA, CPA, and CCA notions of security, and identify a property (reencryption simulatability) that suffices to elevate CPA security to PRE-HRA security.

Claim 1. *Let* PRE *be an PRE-HRA secure proxy reencryption scheme. Then* PRE *is CPA secure.*

[9] Some existing notions in the proxy reencryption literature seem powerful enough to elevate CPA security to PRE-HRA security, including proxy invisibility [AFGH06], unlikability [FL], 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 PRE-HRA security.

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

The proof is below. As we omit a proper definition of CCA security for proxy reencryption, the next claim is stated without proof.[11]

Claim 2. *Let* PRE *be an CCA secure proxy reencryption scheme ([CH07, Definition 2.4*[12]*]). Then* PRE *is PRE-HRA secure.*

Proof of Claim 1. From any probabilistic, polynomial-time algorithm $\mathcal{A}$ (the CPA adversary), we construct an efficient algorithm $\mathcal{A}'$ such that $\mathsf{Adv}^{\mathcal{A}'}_{PRE-HRA}(\lambda) = \mathsf{Adv}^{\mathcal{A}}_{CPA}(\lambda)$; by the PRE-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). For Phase 1, $\mathcal{A}'$ runs Setup and gives $\mathcal{A}$ the generated public parameters; $\mathcal{A}'$ passes all key generation oracle calls and answers unaltered between $\mathcal{A}$ and its own PRE-HRA oracles. $\mathcal{A}'$ begins Phase 2 by requesting all (admissible) reencryption keys $\mathsf{rk}_{i\to j}$ from $\mathcal{O}_{\mathsf{ReKeyGen}}$. Oracle calls from $\mathcal{A}$ to $\mathcal{O}_{\mathsf{ReKeyGen}}$ are answered with the corresponding reencryption key. Oracle 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. Calls from $\mathcal{A}$ to the Challenge and Decision oracles are passed to the corresponding PRE-HRA oracles unaltered. The running time of $\mathcal{A}'$ is polynomially related to the that of $\mathcal{A}$.

Observe that $\mathcal{A}'$ implements the CPA security game perfectly, and $\mathcal{A}$ wins that game if and only if $\mathcal{A}'$ wins the PRE-HRA security game. Therefore $\mathsf{Adv}^{\mathcal{A}'}_{PRE-HRA}(\lambda) = \mathsf{Adv}^{\mathcal{A}}_{CPA}(\lambda)$. □

Reencryption simulatability While PRE-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 PRE-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 secret key sk_i (but with knowledge of the plaintext message $\mathbf{m}$). 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 PRE-HRA security to CPA security. Suppose there existed an adversary $\mathcal{A}_{PRE-HRA}$ violating the PRE-HRA security of a scheme; the reduction plays the roles of both the CPA adversary and the challenger in the PRE-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 $\mathcal{A}_{PRE-HRA}$ to violate CPA security.

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

$$(\mathsf{ReEncSim}(\mathsf{pk}_i, \mathsf{pk}_j, \mathsf{ct}_i, \mathbf{m}), \mathsf{aux}) \approx_s (\mathsf{ReEnc}(\mathsf{rk}_{i\to j}, \mathsf{ct}_i), \mathsf{aux}),$$

[11] As in standard encryption, a number of CCA variants have been studied in the proxy reencryption literature. Using the family of parameterized definitions of [NAL15], Claim 2 holds for $\text{CCA}_{a,2}$ for $a \in \{0,1,2\}$. The claim additionally holds "replayable" variants of CCA security (as is considered in [CH07, LV08]).

[12] More precisely, Definition 2.4 of [CH07] modified to use persistent, rather than ephemeral, reencryption keys, as per Remark 1.

where $\approx_s$ *denotes statistical indistinguishability and* ct_j, ct'_j *and* aux *are sampled according to*

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

A special case of the above is when $\mathsf{ReEncSim}(\mathsf{pk}_i, \mathsf{pk}_j, \mathsf{ct}_i, \mathbf{m}) = \mathsf{Enc}(\mathsf{pk}_j, \mathbf{m})$ simply computes a fresh encryption of the message. That is, if reencrypted ciphertexts are distributed like fresh ciphertexts, then the proxy reencryption is source-hiding.

Theorem 1. *Let* PRE *be an CPA secure, reencryption simulatable, proxy reencryption scheme. Then* PRE *is PRE-HRA secure.*

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

$\mathcal{A}'$ runs $\mathcal{A}$ and simulates the PRE-HRA security game (if $\mathcal{A}$ does not follow the specification of the PRE-HRA security game, $\mathcal{A}'$ simply aborts). To answer oracle calls by $\mathcal{A}$ to any oracle other than $\mathcal{O}_{\mathsf{ReEnc}}$, $\mathcal{A}'$ 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, $\mathcal{A}'$ uses the corresponding reencryption key. On the other hand, for calls to $\mathcal{O}_{\mathsf{ReEnc}}$ from an uncorrupted key to a corrupted key, $\mathcal{A}'$ simulates the reencryption using $\mathsf{ReEncSim}$. This is possible because $\mathcal{A}'$ knows the underlying $\mathbf{m}$. Note that the challenge (for which $\mathcal{A}'$ does not know the underlying plaintext) cannot be reencrypted from uncorrupt to corrupt keys in the PRE-HRA security game.

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

4.2 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 from Section 3.1):

- $\mathsf{KeyGen}_{\mathsf{PRE}}$, $\mathsf{Enc}_{\mathsf{PRE}}$ 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). That is, output $\mathsf{ct}_j = \mathsf{Eval}_{\mathsf{FHE}}(\mathsf{pk}_j, \mathsf{Dec}_{\mathsf{FHE}}, \mathsf{rk}_{i\to j}, \mathsf{ct}_{i\to j})$.

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 PRE-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 from Section 3.1, which is not PRE-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 ct_1,…,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 PRE-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 1, the resulting scheme is therefore PRE-HRA secure.

4.3 Pairings-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, HRsV11].

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 PRE-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 PRE-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 $(Z^{bkk'}, \mathbf{m}Z^{kk'})$.

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

The pairings-based schemes of [ABH09] and [HRsV11] 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 PRE-HRA secure.

References

[ABH09] Giuseppe Ateniese, Karyn Benson, and Susan Hohenberger. Key-private proxy re-encryption. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5473:279–294, 2009.

[ABPW13] Yoshinori Aono, Xavier Boyen, Le Trieu Phong, and Lihua Wang. Key-Private Proxy Re-encryption under {LWE}. (23500031):1–18, 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*, 9(1):1–30, 2006.

[BBS98] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In *Advances in CryptologyEUROCRYPT'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.

[BPRR17] Cristian Borcea, Yuriy Polyakov, Kurt Rohloff, and Gerard Ryan. Picador: End-to-end encrypted publish–subscribe information distribution with proxy re-encryption. *Future Generation Computer Systems*, 71:177–191, 2017.

[CH07] Ran Canetti and Susan Hohenberger. Chosen-Ciphertext Secure Proxy Re-Encryption. pages 1–22, 2007.

[CWYD10] Sherman S M Chow, Jian Weng, Yanjiang Yang, and Robert H Deng. Efficient Unidirectional Proxy Re-Encryption. (60903178):316–332, 2010.

[FL] Xiong Fan and Feng-Hao Liu. Proxy re-encryption and re-signatures from lattices.

[Gen09] Craig Gentry. a Fully Homomorphic Encryption Scheme. *PhD Thesis*, (September):1–209, 2009.

[HHY11] Yi Jun He, Lucas C K Hui, and Siu Ming Yiu. Avoid illegal encrypted DRM content sharing with non-transferable re-encryption. *International Conference on Communication Technology Proceedings, ICCT*, pages 703–708, 2011.

[HRsV11] Susan Hohenberger, Guy N Rothblum, Abhi shelat, and Vinod Vaikuntanathan. Securely Obfuscating Re-Encryption. 24(4):694–719, 2011.

[ID03] Anca Ivan and Yevgeniy Dodis. Proxy cryptography revisited. *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, pages 1–20, 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.

[Kir] Elena Kirshanova. Proxy Re-encryption from Lattices. pages 77–94.

[LPK10] Sangho Lee, Heejin Park, and Jong Kim. A secure and mutual-profitable DRM interoperability scheme. *Proceedings - IEEE Symposium on Computers and Communications*, pages 75–80, 2010.

[LV08] Benoit Libert and Damien Vergnaud. Unidirectional Chosen-Ciphertext Secure Proxy Re-encryption. pages 360–379, 2008.

[NAL15] David Nunez, Isaac Agudo, and Javier Lopez. A Parametric Family of Attack Models for Proxy Re-encryption. *Proceedings of the Computer Security Foundations Workshop*, 2015-Septe(Csf):290–301, 2015.

[PRSV17] Yuriy Polyakov, Kurt Rohloff, Gyana Sahu, and Vinod Vaikuntanthan. Fast proxy re-encryption for publish/subscribe systems. 2017.

[PWA$^+$16] Le Trieu Phong, Lihua Wang, Yoshinori Aono, Manh Ha Nguyen, and Xavier Boyen. Proxy Re-Encryption Schemes with Key Privacy from LWE. *IACR Cryptology ePrint Archive*, 2016:327, 2016.