A preliminary version of this paper will appear at CRYPTO 2017; this is the full version (June 2, 2017).

Hedging Public-Key Encryption in the Real World

Alexandra Boldyreva,[1] Christopher Patton,[2] and Thomas Shrimpton[2]

[1] Georgia Institute of Technology
[2] University of Florida

Abstract. Hedged PKE schemes are designed to provide useful security when the per-message randomness fails to be uniform, say, due to faulty implementations or adversarial actions. A simple and elegant theoretical approach to building such schemes works like this: Synthesize fresh random bits by hashing all of the encryption inputs, and use the resulting hash output as randomness for an underlying PKE scheme. The idea actually goes back to the Fujisaki-Okamoto transform for turning CPA-secure encryption into CCA-secure encryption, and is also used to build deterministic PKE schemes.

In practice, implementing this simple construction is surprisingly difficult, as the high- and mid-level APIs presented by the most commonly used crypto libraries (e.g. OpenSSL and forks thereof) *do not* permit one to specify the per-encryption randomness. Thus application developers are forced to piece together low-level functionalities and attend to any associated, security-critical algorithmic choices. Other approaches to hedged PKE present similar problems in practice.

We reconsider the matter of building hedged PKE schemes, and the security notions they aim to achieve. We lift the current best-possible security notion for hedged PKE (IND-CDA) from the CPA setting to the CCA setting, and then show how to achieve it using primitives that are readily available from high-level APIs. We also propose a new security notion, MM-CCA, which generalizes traditional IND-CCA to admit imperfect randomness. Like IND-CCA, and unlike IND-CDA, our notion gives the adversary the public key. We show that MM-CCA is achieved by RSA-OAEP in the random-oracle model; this is significant in practice because RSA-OAEP is directly available from high-level APIs across all libraries we surveyed. We sort out relationships among the various notions, and also develop new results for existing hedged PKE constructions.

Keywords: hedged public-key encryption, cryptographic APIs

Table of Contents

1 Introduction

The security of many cryptographic primitives relies on access to reliable, high-quality randomness. However, generating good randomness is a complex process that often fails, due to use of ill-designed random number generators (RNGs), software bugs, or malicious subversion [GM05, Mue08, DGP09, DPR$^+$13, MMS13, CMG$^+$16]. Such failures have led to serious breaches of security in deployed cryptographic schemes [RY10, HDWH12, BCC$^+$13, CMG$^+$16]. Recent high-profile examples include security vulnerabilities in a significant fraction of TLS and SSH servers caused by problems with RNGs as exposed by Heninger et al. [HDWH12] and the vulnerabilities with Juniper NetScreen-branded firewalls that use Dual EC RNG designed by NSA to have a backdoor, as studied by Checkoway et al. in [CMG$^+$16].

Theorists have begun to address the practical issue of weak randomness. Of particular interest has been the case of public-key encryption (PKE), since there are no shared secrets upon which to bootstrap security. In their seminal work [BBN$^+$09], Bellare et al. introduce the notion of hedged public-key encryption. Informally, hedged encryption guarantees traditional semantic security when the per-message randomness is perfect, and retains best-possible security guarantees when not, assuming there is sufficient min-entropy in the joint distribution over the plaintext messages and the per-message randomness. Such security is called hedged security.

A particularly simple and elegant approach to building hedged PKE is what Bellare et al. refer to as Encrypt-with-Hash (EwH)[3]. Loosely, to encrypt a message M (and potentially some auxiliary input I) using public key pk and randomness r, one computes a string $\tilde{r}$ by hashing (pk, M, I, r), and then returns a ciphertext $\mathcal{E}(pk, M; \tilde{r})$. In the random oracle model (ROM) [BR93], any entropy contained among the hash inputs is harvested to synthesize new randomness $\tilde{r}$ that can be treated as uniform. Intuitively, unless the attacker manages to guess (pk, M, I, r), or $\tilde{r}$ directly, this EwH scheme remains hedged-secure if the underlying scheme $\mathcal{E}$ is IND-CPA.

Other works on hedged PKE and related efforts to deal with imperfect per-message randomness have followed this approach [Yil10, RSV13, PSS14, BT16]. It has also been used to construct deterministic encryption [BBO07, BFO08, RSV13]. In fact, this trick of synthesizing randomness for encryption dates back (at least) to Fujisaki and Okamoto [FO99], who used this as part of a transform to turn CPA-secure encryption into CCA-secure encryption.

EwH in practice. Say that a developer is aware of the security breaches caused by bad randomness, and wants to implement EwH using the best-known and most widely-deployed cryptographic library, OpenSSL. To protect application developers from having to understand and properly handle lower-level algorithmic details, OpenSSL encourages the use of high-level "envelope" API calls. For public-key encryption, the interface is

```
int EVP_PKEY_encrypt(EVP_PKEY_CTX *ctx, unsigned char *out,
     size_t *outlen, const unsigned char *in, size_t inlen)
```

where `ctx` points to the so-called encryption context, which acts as state across calls. Among other things, it contains the public key and a descriptor of the particular PKE scheme to be used: Textbook RSA, PKCS #1 v1.5 RSA encryption (RFC 2313), and a variant of RSA-OAEP [BR95] specified in PKCS #1 v2.2 (RFC 8017). The plaintext input is pointed to by `in`, and `out` points to where the ciphertext output should be written. Notice: *Nowhere is one able to specify the randomness to be used.* The mid-level function calls that are wrapped by `EVP_PKEY_encrypt` also do not expose the

[3] To be precise, [BBN$^+$09] refers to their constructions as REwH, and those are extensions of the EwH scheme from [BBO07]. We use the name EwH for simplicity.

randomness to the caller. One could try to manipulate the source of randomness, `RAND_bytes`, used by the higher-level calls. Indeed, OpenSSL provides an interface for adding entropy into the state of the underlying (P)RNG; doing so, however, presents several technical challenges, which we discuss at length in Section 2. Hence, to implement EwH in OpenSSL, the developer is forced to cobble together low-level functionalities, which implies needing to attend to security-critical details, such as parameters, padding schemes, or how the randomness is generated. The same is true for the two most popular forks of OpenSSL (BoringSSL and LibreSSL) and several other common libraries. We give a survey of crypto libraries in Section 2.

Encrypt-with-Hash is not the only approach to building hedged PKE (or deterministic PKE, etc.), and we will discuss some others shortly. But the punchline there will be the same: Developers face similar hurdles when they attempt to instantiate those constructions with modern crypto libraries.

To summarize, while hedged PKE has received significant theoretical study, the gap between theory and practice remains large. Existing theoretical constructions offer little to developers who respect the guidance of widely deployed crypto libraries to use high-level APIs.

Reconsidering hedged PKE. We reconsider the matter of constructing PKE schemes that maintain useful security guarantees when forced to use imperfect randomness. There are two important questions that guide us:

- What simple and efficient schemes can we implement via high-level APIs exported by standard crypto libraries?
- What security notions can we hope to achieve with these schemes?

To the latter question, we take as our starting point the IND-CDA notion of [BBN+09], which we rename as MMR-CPA. In the MMR-CPA experiment, the adversary may query an encryption oracle with sources $\mathcal{M}$, each of these outputting a triple $(\boldsymbol{M}_0, \boldsymbol{M}_1, \boldsymbol{r})$, consisting of a pair of vectors of messages and a vector of randomness to be used for encryption (hence MMR). The oracle, which contains the public key pk and a secret challenge bit b, returns a vector of component-wise encryption of $\boldsymbol{M}_b$, each under the corresponding component randomness from $\boldsymbol{r}$. The adversary's goal is to guess the value of b. Crucially, the adversary is not provided with the public key pk until after all encryption queries are made; otherwise, pk-dependent $\mathcal{M}$ can be crafted that would make MMR-CPA unachievable, even when $\mathcal{M}$ is a high min-entropy source [BBN+09]. Also implicit is that the public key was generated using uniform coins, and that only the per-message randomness is under suspicion.

Achieving MMR-CCA. As a small definitional contribution, we extend MMR to the CCA setting, and both the CPA and CCA notions are formalized for PKE with associated data (AD). Associated data was originally called "labels" in the PKE literature [Sho04, DK05, CCS09, ABP16]. But AD seems to be more often used among practitioners, so we adopt it. (This also aligns better with the language of symmetric encryption.)

The MMR attack effectively assumes the adversary can arbitrarily and adaptively re-corrupt the randomness source used by the libraries when producing ciphertexts. In many settings, where the per-message randomness source is provided by the operating system (or even hardware), this equates to re-corrupting the OS (or hardware) at will with each encryption. The strength of this attack model makes RSA-OAEP, for example, unable to achieve MMR-CPA (let alone -CCA)

Construction	**Assumptions**	**Achieves**
F-EME-OAEP	F is POWF	MM+IND-CCA
HE$[F, \text{AEAD}]$	F is OWF, AEAD is IND-CPA+AUTH	MMR+IND-CCA
PtD$[F\text{-DOAEP}]$	F is OWF	MM+IND-CCA
RtD$[\Pi_r, F\text{-DOAEP}]$	Π_r is IND-CPA, F is OWF	MM+IND-CPA

Fig. 1. A summary of our constructions and the security they achieve.

security.[4] This is unfortunate, as RSA-OAEP is the only provably-secure scheme implemented by `EVP_PKEY_encrypt`, and it is available across virtually all libraries. In fact, there are currently *no* positive results for RSA-OAEP in the presence of imperfect randomness.

That said, we give the first MMR-CCA secure PKE scheme. It is a hybrid-encryption construction that uses a trapdoor function, a hash function (modeled as a random oracle), and a symmetric-key authenticated encryption scheme. Each of these components can be called with most crypto libraries, including OpenSSL, via high-level APIs. We prove that the scheme is MMR-CCA in the ROM assuming the standard assumptions on security of the base schemes. Despite the simplicity of the scheme, the security proof is quite involved. See Section 6.2 for details.

The MM notions. The MMR notions define security in the hedged PKE setting with imperfect randomness, yet no common crypto library explicitly exposes a single primitive that achieves it. We define a new pair of notions, MM-{CPA,CCA}, which are identical to their MMR counterparts but with two important exceptions. First, the adversary is provided the public key as initial input. Second, the per-message randomness source $\mathcal{R}$ may be corrupted *once*, prior to any encryptions. This models scenarios in which the OS code base, a standards document, or a hardware RNG may have been modified (maliciously or otherwise) to produce faulty randomness prior to widespread distribution. And, while it is good practice to be cautious, we are unaware of any practical scenarios or documented attacks in which the randomness source may be continuously re-corrupted to depend on previously observed ciphertexts and the messages about to be encrypted, as is allowed in the MMR attack setting.

We show that RSA-OAEP is MM-CCA secure (in the ROM) whenever $\mathcal{R}$ has min-entropy sufficient to stop attacks that would break *any* PKE scheme in the MM setting. Not only does this give the first positive result for RSA-OAEP in the presence of imperfect randomness, but it also gives developers an immediate option across virtually all libraries.

Because MM adversaries are given the public key, MM security against adaptive attackers follows "for free" (via a standard hybrid argument) from MM security against non-adaptive attackers. On the other hand, in general one converts non-adaptive MMR security into adaptive MMR security only with the addition of an extra key-anonymity property (ANON); Bellare et al. [BBN+09] show this in the CPA setting, and we give an analogous result in the CCA setting.

Relating the notions. We view MM-{CPA,CCA} as a direct generalization of IND-{CPA,CCA}. In the latter, the randomness source is perfect, and the adversary queries (effectively) a source whose support contains exactly one pair (M_0, M_1), i.e., a source with zero min-entropy. We work out relationships among the MM, MMR and IND notions. Among them, we show that IND-CCA $\not\Longrightarrow$ MM-CCA in general, which makes our positive result for RSA-OAEP non-trivial.

[4] Consider the plaintext-recovery attack by Brown [Bro05] on RSA-OAEP with public exponent $e = 3$. The attack exploits low entropy coins and is effective even if messages have high min-entropy.

Perhaps unintuitively, we show that the MMR notions are *not* stronger security notions than the MM notions. They are incomparable: in the MMR setting, the adversary is allowed to re-corrupt the randomness source but does not have the public key; in the MM setting, the adversary has the public key, but may only use it to produce message sources, and may not re-corrupt the randomness source.

Hedging beyond EwH. Not all previous proposals for hedged encryption require direct manipulation of the randomness used by some underlying PKE scheme. For example, Bellare et al. [BBN+09] propose doing $\mathcal{E}_d(pk_d, \mathcal{E}_r(pk_r, M;\, r))$, which first encrypts the message M using a randomized PKE scheme $\mathcal{E}_r$, and then re-encrypts the resulting ciphertext using a deterministic scheme. They call this the Randomized-then-Deterministic (RtD) composition. (Note that this means two public-keys are needed, potentially requiring the issuing of new certificates, among other deployment issues.) They also propose a construction called Pad-then-Deterministic (PtD), where $\mathcal{E}(pk_d, M)$ is defined by sampling randomness r and then returning $\mathcal{E}_d(pk_d, M \,\|\, r)$. In both cases, to provide security against weak randomness, it is necessary (although not sufficient) that the deterministic scheme is PRIV-secure in the sense of [BBO07].

Here, too, we run into problems in practice. Standard crypto libraries do not offer function calls that directly implement any PRIV-secure deterministic PKE schemes. Several such schemes are known in the literature [BBO07, BBN+09, BFO08, RSV13], but implementing these would require piecing together calls to low-level functionalities, precisely what modern APIs attempt to avoid.

One potential exception is RSA-DOAEP [BBO07], a three-round Feistel construction followed by a single call to RSA. This is the most amenable scheme to being implemented from high-level calls — OpenSSL exposes `EVP` calls for hashing, and the `EVP_PKEY_encrypt` function admits raw RSA as one of its options.

We show that RtD, where the deterministic scheme is DOAEP, is both MM-CPA and IND-CPA secure. Better yet, we are able to show, under appropriate conditions, that PtD with DOAEP is MM+IND-CCA secure.

Open questions. Our work leaves open some interesting questions. For one, is MM-CCA achievable in the standard model? In particular, from reasonable assumptions and via primitives that are available in crypto libraries (without making very low-level calls)? Asking a bit less, is MM-CPA achievable with the same restrictions? By composing two of the theorems we give, any scheme that is (non-adaptive) MMR-CCA and ANON-CCA in the standard model would be MM-CCA, too. But this only shifts the focus to the question of how to build schemes that achieve these two properties, and within the constraints we mentioned.

In an analogous result, we show that a scheme that is (non-adaptive) MMR-CPA and ANON-CPA in the standard model is MM-CPA. Prior work does give schemes that are non-adaptive MMR-CPA and ANON-CPA (e.g., the RtD and PtD schemes from [BBN+09]), but none that can be realized from typical high-level APIs. So from our perspective, achieving MM-CPA in the standard model remains open in practice.

A call to action. A theoretician's viewpoint on this work might be to suggest that libraries should be modified to keep up with the nice primitives that our community provides. In practice, this viewpoint is unhelpful. The design of good APIs, like the design of good cryptography, is hard work. A recent study by Acar et al. [ABF+17] reveals that modern APIs make even simple tasks difficult to implement, which has been shown time and time again to result in security vulnerabilities in real systems. Yet, the question of what is the "right" level of exposure to the user is a complex trade-off between usability and flexibility. APIs have very long lifetimes because, once adopted, changing them potentially implies altering all of the applications upon which they are built. Our thesis is that raising awareness of real APIs in our research community will better serve cryptographic practice, and will uncover interesting new theory challenges (like those we explore) as well.

Related work. Raghunathan et al. [RSV13] extend the security notion for deterministic encryption to the setting where the adversary is given the public key. They also consider chosen-ciphertext attacks and argue that their extension can be applied to hedged encryption. So that their notion is achievable, the adversary is restricted to choosing sources for its queries from a finite set (whose size is bounded by a parameter of the experiment) of sources that do not depend on the public key. We note a similar restriction in the MM-CCA setting; the randomness source may not depend on the public key, since otherwise the source could be crafted to leak information about the plaintext. Their definition is incomparable to our MM-CCA notion, and it is not clear what practical threat model it captures. Moreover, their definition deems RSA-OAEP insecure, while our MM-CCA definition permits for useful security analysis of the most deployed PKE scheme, in case of imperfect randomness.

Paterson et al. [PSS14] give notions of security under related-randomness attacks (RRA). Here, too, the adversary is provided with the public key. The RRA notions generalize the reset attack (RA) notions due to Yilek [Yil10] by allowing the adversary to specify certain functions to be applied to fresh uniform randomness, or to previously sampled uniform randomness, and have the result used to encrypt chosen plaintexts. These functions must be output-unpredictable, loosely meaning that they cannot allow the attacker to guess the randomness that will be used for encryption, and collision-resistant, meaning that the queried functions, if applied to the same uniform random string, should not produce the same output. If either of these conditions is violated, there is an attack that makes RRA security impossible for any scheme. This is similar to our requirement in the MM notions that the encryption randomness have min-entropy that is $\omega(\log k)$, where k is the security parameter. Again, their definition is incomparable to our MM-CCA notion, and unlike our definition, does not allow to consider randomness sources with arbitrary high-min-entropy distributions. We note that again, RRA security is not achievable by randomness-recovering PKE schemes, such as RSA-OAEP.

Bellare and Tackmann [BT16] give notions of hedged security in the presence of nonces. They consider a setting where a sender uses a uniform seed and a nonce, and security is guaranteed if either the seed is secret and the nonces are non-repeating, or the seed is compromised and the nonces are unpredictable. Brzuska et al. and Bellare and Hoang [BFM15,BH15] show that assuming the existence of indistinguishability obfuscation (iO), the random oracle in the EwH construction is uninstantiable. Finally, Hoang et al. [HKOZ16] study public-key encryption security against selective-opening attacks in the presence of randomness failures.

2 Crypto libraries

In this section we provide a brief survey of real-world libraries: In particular, the extent to which their APIs for PKE expose the per-message encryption randomness. A table with Web links for each library is found in Appendix B.

We begin with OpenSSL, the most widely-used library for encryption on the Web. As discussed in the introduction, OpenSSL encourages the use of “envelopes”, which are designed to abstract the details of the algorithm used. We have noted that the high-level call `EVP_PKEY_encrypt` does not allow the programmer to specify the source of entropy. This call is a wrapper for RSA-based encryption, internally invoked by calling `RSA_public_encrypt`. This function has the interface

```
int RSA_public_encrypt(int flen, unsigned char *from,
    unsigned char *to, RSA *pk, int padding)
```

It allows one to specify one of three padding schemes (via `padding`), which is passed down from the `ctx` input of `EVP_PKEY_encrypt`. So we see that here, too, there is no explicit place to insert external randomness. Similarly, the functions `EVP_SealInit`, `EVP_SealUpdate`, and `EVP_SealFinal`, which provide a high-level abstraction for hybrid encryption, do not surface the coins.

This design pattern is maintained by BoringSSL and LibreSSL, the two most popular forks of the OpenSSL codebase. It is also adopted by a number of other libraries, including the popular open source libraries libgcrypt and PyCrypto, as well as the commercial library cryptlib.

The *SSL API style reflects the opinion that APIs should not allow application developers to touch the coins, as doing so invites errors that can fatally impact security. Indeed, at Real World Cryptography 2017, Google security-team developers said interfaces should "Never ask users to provide critical input (e.g., randomness, etc.)." [DKN].

The commercial library cryptlib exposes three levels of abstraction, each intended for a different audience. The highest level provides envelopes and secure sessions that expose no cryptography at all; the middle level exposes the notion of keys and signatures, offering more flexibility for protocol designers, but does not expose algorithms; and finally, the lowest level exposes actual algorithms, but even this API does not allow the programmer to specify the coins.

Hedging via providing the coins source. Of course, there are APIs that surface access to the coins directly. For example, in Go's native crypto library the function call for RSA-OAEP has the signature

```
func EncryptOAEP(hash hash.Hash, random io.Reader,
    pub *PublicKey, msg []byte, label []byte)
```

The randomness source is the second parameter of this routine. One can hedge RSA-OAEP by implementing the `io.Reader` interface. Other examples of APIs that expose the coins are Botan, Crypto++, wolfSSL, and SCAPI.

Falling (somewhat confusingly) in the middle is the popular Java library known as Bouncy Castle. Java provides a built-in interface for various security-related functionalities. The programmer can control which library implements these functionalities by specifying a *security provider*, e.g., Bouncy Castle. Bouncy Castle's own API does *not* surface coins. On the other hand, the native Java API does. For instance, one initializes a structure for ElGamal encryption [Elg85] as follows. Let `pubKey` be an ElGamal public key:

```
Cipher cipher = Cipher.getInstance("ElGamal/None/NoPadding", "BC");
cipher.init(Cipher.ENCRYPT_MODE, pubKey, new SecureRandom());
```

The string `"BC"` means the security provider is Bouncy Castle. So one could instantiate EwH (over ElGamal) here by providing their own implementation of `SecureRandom`.

Hedging via reseeding the coins source. Although OpenSSL does not explicitly surface the coins, it exposes an interface for manipulating the coins used to provide randomness for higher-level calls. Coins are sampled in OpenSSL via the interface `RAND_bytes(unsigned char *buf, int num)`, which writes the next `num` bytes output by the source to `buf`. By default, the output is a stream of bytes generated by a PRNG seeded with entropy gathered by the system, e.g., via `/dev/urandom` . When the PRNG is called, it generates the requested bytes and updates its internal state by applying a cryptographic hash function. (The hash function may be specified by the programmer.) Alternatively, a hardware-based RNG can be used. For our purposes, there are two relevant ways to manipulate the state:

– `RAND_seed(const void *buf, int num)`: Resets the state using the first `num` bytes of `buf` as a seed.
– `RAND_add(const void *buf, int num, double entropy)`: "Mixes" the first `num` bytes of `buf` into the state. `entropy` is an estimate of the number of full bytes of entropy of the input.

A search of the source code[5] reveals that the implementation of the padding scheme calls `RAND_bytes`. To hedge RSA-OAEP using this interface, one might do as follows:

```
RAND_add((const void *)in, in_len, in_entropy);
ctxt_len = RSA_public_encrypt(msg_len, msg, ctxt, pk,
                              RSA_PKCS1_OAEP_PADDING);
```

where `in_entropy` is an estimate of the bytes of entropy of the string `in`, which encodes `pk`, and `msg`. There are a number of technical details to attend to here. First, estimating the entropy of `in` is non-trivial. (The OpenSSL documentation refers the reader to RFC 1750 for estimation methodologies.[6]) Second, the documentation does not specify how the state is updated, except that if `entropy` $=$ `num` , then this call is equivalent to resetting the state via `RAND_seed`, effectively evicting the initial entropy provided by the system. Third, if a hardware RNG is used to instantiate `RAND_bytes`, then calling `RAND_add` fails silently, meaning *the call has no effect on the randomness.* Alternatively, one might first call `RAND_bytes(rand, rand_len)`, then reset the state via `RAND_seed` on input of a buffer containing `pk`, `msg`, and `rand`. Again, if a hardware RNG is used, then calling `RAND_seed` has no effect.

Apart from these practical considerations, we note a subtle theoretical issue with hedging OpenSSL in this manner. At first glance, it would appear that if one is careful with the technical details, then these interfaces could be used to implement EwH. However, since the PRNG is stateful, the coins used to encrypt a message necessarily depend on the inputs of all prior encryptions. It is not clear that the proof security for EwH holds for this instantiation, since the message-coins source is assumed to be stateless [BBN+09, Theorem 6.1].

To summarize, if a developer chooses to (or must) use a library whose APIs do not expose the encryption randomness, e.g., any of the widely-deployed *SSL libraries, they are forced to work with low-level functionalities and attend to security-critical details about parameters, padding, the implementation of the (P)RNG, etc. If they are free to work with, say, the Go native library, then they can implement EwH by extending the functionality of the exposed randomness source.

3 Preliminaries

Notation. If n is an integer we write $[n]$ for the set $\{1, 2, \ldots, n\}$. If i and j are integers such that $i \leq j$, we let $[i..j]$ denote the set $\{i, i+1, \ldots, j\}$. (If $i > j$, then let $[i..j] = \emptyset$.) The implicit, unambiguous encoding of one or more objects as a bit string is written as $\langle X, Y, \ldots \rangle$. We write vectors in boldface, e.g., $\boldsymbol{X}$. We let $\boldsymbol{X}_i$ and $\boldsymbol{X}[i]$ denote the i-th element of $\boldsymbol{X}$. We say that $\boldsymbol{X}, \boldsymbol{Y}$ are *length-equivalent* if $|\boldsymbol{X}| = |\boldsymbol{Y}| = m$ and, for all $i \in [m]$, $|\boldsymbol{X}_i| = |\boldsymbol{Y}_i|$. We let Λ denote the empty vector. All algorithms, unless noted otherwise, are randomized. An *adversary* is a randomized algorithm. The runtime of adversary $\mathcal{A}$ (at security parameter k) is denoted $\mathsf{time}_{\mathcal{A}}(k)$.

Games. We adopt the game-playing framework of Bellare and Rogaway [BR06]. The notation $\mathbf{Exp}(\mathcal{A}, k)$ denotes the execution of game $\mathbf{Exp}$ with adversary $\mathcal{A}$ at security parameter k. Let $\mathbf{Exp}(\mathcal{A}, k) \Rightarrow x$ be the random variable denoting the event that game $\mathbf{Exp}$ outputs x when played

[5] See `https://github.com/openssl/openssl/blob/OpenSSL_1_0_2-stable/crypto`.
[6] See `https://wiki.openssl.org/index.php/Manual:RAND_add(3)`.

by $\mathcal{A}$ at security parameter k. If the outcome of the game is either true or false, then we write $\mathbf{Exp}(\mathcal{A}, k)$ as short hand for $\mathbf{Exp}(\mathcal{A}, k) \Rightarrow \mathsf{true}$.

3.1 Public-key encryption with associated data

A public-key encryption scheme with associated data PKEAD is a triple of algorithms $(\mathsf{Kgen}, \mathsf{Enc}, \mathsf{Dec})$ with associated data space $\mathsf{AD} \subseteq \{0,1\}^*$ and randomness length $\rho(\cdot)$. The *key-generation algorithm* Kgen takes 1^k as input, and outputs a pair of strings (pk, sk), the public key and secret key respectively. The *encryption algorithm* takes as input the public key pk, associated data $H \in \mathsf{AD}$, message $M \in \{0,1\}^*$, and coins $r \in \{0,1\}^{\rho(k)}$ and outputs a ciphertext $C \in \{0,1\}^*$ or the distinguished symbol $\perp$, indicating that encryption failed. When the value of the coins used is not important, we write $\mathsf{Enc}(pk, H, M)$ or $\mathsf{Enc}^H_{pk}(M)$ as short hand for $r \leftarrow\$ \{0,1\}^{\rho(k)}; \mathsf{Enc}(pk, H, M; r)$. Otherwise, we write $\mathsf{Enc}(pk, H, M; r)$ or $\mathsf{Enc}^H_{pk}(M; r)$. The *decryption algorithm* takes the secret key sk, associated data $H \in \mathsf{AD}$, and a ciphertext $C \in \{0,1\}^*$ and outputs a message $M \in \{0,1\}^*$ or $\perp$, indicating failure to decrypt. Just as for encryption, we write $M \leftarrow \mathsf{Dec}(sk, H, C)$ or $M \leftarrow \mathsf{Dec}^H_{sk}(C)$.

It will be convenient to define vector-valued encryption. To that end, let $v \in \mathbb{N}$, $\boldsymbol{M} \in (\{0,1\}^*)^v$, and $\boldsymbol{H} \in \mathsf{AD}^v$. Then the notation $\boldsymbol{C} \leftarrow\$ \mathsf{Enc}(pk, \boldsymbol{H}, \boldsymbol{M})$ means to compute $\boldsymbol{C}_i \leftarrow\$ \mathsf{Enc}(pk, \boldsymbol{H}_i, \boldsymbol{M}_i)$ for every $i \in [v]$, and to assemble $\boldsymbol{C} = (\boldsymbol{C}_1, \ldots, \boldsymbol{C}_v)$ as the return value.

In this work, we consider schemes for which the following holds: If for every $k \in \mathbb{N}$, $(pk, sk) \in [\mathsf{Kgen}(1^k)]$, $H \in \mathsf{AD}$, and $M \in \{0,1\}^*$, there exists an $r' \in \{0,1\}^{\rho(k)}$ such that $\mathsf{Enc}^H_{pk}(M; r') \neq \perp$, then for every $r \in \{0,1\}^{\rho(k)}$, it holds that $\mathsf{Enc}^H_{pk}(M; r) \neq \perp$. Such a scheme is *correct* if for every $k \in \mathbb{N}$, $(pk, sk) \in [\mathsf{Kgen}(1^k)]$, $H \in \mathsf{AD}$, $M \in \{0,1\}^*$ and $r \in \{0,1\}^{\rho(k)}$, we have $C \neq \perp \implies \mathsf{Dec}^H_{sk}(C) = M$, where $C = \mathsf{Enc}^H_{pk}(M; r)$. As this condition makes clear, proper operation is demanded when both encryption and decryption are in possession of H. We note H may be the empty string, recovering more traditional public-key encryption.

Note that our formulation of public-key encryption deviates from tradition in that it allows for associated data. Associated data has already been considered in formalizations of asymmetric encryption, e.g., [Sho04, DK05, CCS09, ABP16], where it was called a "label". We prefer to call it associated data to better align with the symmetric setting, where associated data has become a commonplace term. There are settings in which it is desirable to consider associated data in the public-key setting, for example, to enforce policies on the use of the resulting plaintext. Moreover, one of the constructions we propose is from an AEAD scheme, so it is natural to surface the associated data as the input of the PKEAD scheme.

3.2 Sources

In our security definitions, we will rely on the notion of a *source*, so we start with generalizing this notion as described in [BBN+09]. Let β and γ be non-negative integers, k be a positive integer, and $\mu, v, \rho_0, \ldots, \rho_{\gamma-1} : \mathbb{N} \to \mathbb{N}$ be functions. We define a $(\mu, v, \rho_0, \rho_1, \ldots, \rho_{\gamma-1})$*-*$\mathrm{m}^\beta\mathrm{r}^\gamma$*-source* $\mathcal{M}$ as an algorithm that on input 1^k returns a tuple $(\boldsymbol{M}_0, \boldsymbol{M}_1, \ldots, \boldsymbol{M}_{\beta-1}, \boldsymbol{r})$ with the following properties: one, for every $b \in [0..\beta-1]$, vector $\boldsymbol{M}_b$ is over strings; two, vector $\boldsymbol{r}$ is over γ-tuples of strings; three, each of the vectors has $v(k)$ elements; four, for every $i \in [v(k)]$ and $c \in [0..\gamma-1]$, string r_c has length $\rho_c(k)$ where $(r_0, \ldots, r_{\gamma-1}) = \boldsymbol{r}[i]$; five, for every $b, b' \in [0..\beta-1]$, vectors $\boldsymbol{M}_b$ and $\boldsymbol{M}_{b'}$ are length-equivalent; and six, for every $k \in \mathbb{N}$, $b \in [0..\beta-1]$, $i \in [v(k)]$, and $(M, r) \in \{0,1\}^{|\boldsymbol{M}_b[i]|} \times (\{0,1\}^{\rho_0(k)} \times \cdots \times \{0,1\}^{\rho_{\gamma-1}(k)})$ it holds that

$$\Pr\left[\, (\boldsymbol{M}_0, \ldots, \boldsymbol{M}_{\beta-1}, \boldsymbol{r}) \leftarrow\$ \mathcal{M}(1^k) : (\boldsymbol{M}_b[i], \boldsymbol{r}[i]) = (M, r) \,\right] \leq 2^{-\mu(k)}.$$

We say that such a source has output length $v(\cdot)$ and min-entropy $\mu(\cdot)$. When stating the parameters is not important, we refer to the source as an $\mathrm{m}^\beta \mathrm{r}^\gamma$-source. In this paper we will consider mr-, mmr-, mm-, and r-sources.

We define the *equality pattern* of $v(k)$-vectors $\boldsymbol{M}$ and $\boldsymbol{r}$ as the bit-valued matrix $\mathrm{E}^{\boldsymbol{M},\boldsymbol{r}}$ defined by $\mathrm{E}^{\boldsymbol{M},\boldsymbol{r}}[i,j] = 1 \iff (\boldsymbol{M}[i], \boldsymbol{r}[i]) = (\boldsymbol{M}[j], \boldsymbol{r}[j])$ for every $i,j \in [v(k)]$. A $(\mu, v, \rho_0, \ldots, \rho_{\gamma-1})$-$\mathrm{m}^\beta \mathrm{r}^\gamma$-source is *distinct* if for every $k \in \mathbb{N}$ and $b \in [0..\beta{-}1]$, it holds that $\Pr[(\boldsymbol{M}_0, \ldots, \boldsymbol{M}_{\beta-1}, \boldsymbol{r}) \leftarrow\!\!\$\, \mathcal{M}(1^k) : \mathrm{E}^{\boldsymbol{M}_b,\boldsymbol{r}} = \mathrm{I}_{v(k)}] = 1$, where $\mathrm{I}_{v(k)}$ denotes the $v(k) \times v(k)$ identity matrix. Security against chosen distribution attacks will be defined with respect to adversaries that specify distinct sources. We remark that it is possible to relax this requirement somewhat [BBN+09, section 4.3], but we will not belabor this point.

4 Security notions

Let $\mathsf{PKEAD} = (\mathsf{Kgen}, \mathsf{Enc}, \mathsf{Dec})$ be a PKEAD scheme with associated data space AD and randomness length $\rho(\cdot)$. (We will refer to PKEAD throughout this section.) In this section we define three notions of privacy. The first, IND-CCA, is standard (IND-CCA2 in the taxonomy of [BDPR98]), except that it considers associated data. In this notion, the source of coins for encryption is fixed and uniform. The second, MMR-CCA is a lifting of the MMR-CPA notion from [BBN+09] (where it is called IND-CDA) to the CCA setting with associated data. In this notion, the adversary is free to re-corrupt the source of coins on each encryption. The third, MM-CCA, is entirely new. In this notion, the coins source is corrupted once prior to the keys being chosen and any encryption are made. We now discuss the notions (presented in Figure 2) in more detail. For each attack and setting $(\mathrm{ATK}, \mathrm{STG}) \in \{\mathrm{IND}, \mathrm{MMR}, \mathrm{MM}\} \times \{\mathrm{CPA}, \mathrm{CCA}\}$ we define $\mathbf{Adv}^{\text{atk-stg}}_{\mathsf{PKEAD}}(\mathcal{A}, k) = 2 \cdot \Pr\left[\mathbf{Exp}^{\text{atk-stg}}_{\mathsf{PKEAD}}(\mathcal{A}, k)\right] - 1$.

4.1 IND security

The standard notion of indistinguishability under chosen-ciphertext attacks is generalized to incorporate associated data in Figure 2. We say that PKEAD is IND-CCA secure if for every PT ("polynomial-time") adversary $\mathcal{A}$, the function $\mathbf{Adv}^{\text{ind-cca}}_{\mathsf{PKEAD}}(\mathcal{A}, \cdot)$ is negligible. The corresponding notion in the chosen-plaintext attack setting is obtained by denying the adversary access to the decryption oracle. Let $\mathbf{Exp}^{\text{ind-cpa}}_{\mathsf{PKEAD}}(\mathcal{A}, k)$ denote this experiment. We say that PKEAD is IND-CPA secure if for every PT adversary $\mathcal{A}$, the function $\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{PKEAD}}(\mathcal{A}, \cdot)$ is negligible.

4.2 MMR security

We adapt the definition of security against chosen-distribution attacks (IND-CDA) from [BBN+09] to deal with associated data and chosen-ciphertext attacks.

Consider the MMR-CCA experiment defined in Figure 2 associated to PKEAD, adversary $\mathcal{A}$, and security parameter k. The output of the $\mathbf{LR}$ oracle is well-defined if for every $k \in \mathbb{N}$ and some $\mu, v : \mathbb{N} \to \mathbb{N}$, it holds that $\mathcal{M}$ is a (μ, v, ρ)-mmr-source, and $\boldsymbol{H} \in \mathsf{AD}^{v(k)}$. Fix functions $\mu, v : \mathbb{N} \to \mathbb{N}$ where $\mu(k) \in \omega(\log k)$. We call $\mathcal{A}$ a (μ, v, ρ)-mmr-adversary if its queries are well-defined and its $\mathbf{LR}$ queries consist of distinct (μ, v, ρ)-mmr-sources. We say that PKEAD is MMR-CCA secure with respect to distinct (μ, v, ρ)-mmr-sources if for every polynomial-time (μ, v, ρ)-mmr-adversary $\mathcal{A}$, the function $\mathbf{Adv}^{\text{mmr-cca}}_{\mathsf{PKEAD}}(\mathcal{A}, \cdot)$ is negligible.

```
$\mathbf{Exp}^{\text{ind-cca}}_{\mathsf{PKEAD}}(\mathcal{A}, k)$:
$Q \leftarrow \emptyset$
$(pk, sk) \leftarrow\$ \mathsf{Kgen}(1^k)$
$b \leftarrow\$ \{0, 1\}$
$b' \leftarrow\$ \mathcal{A}^{\mathbf{LR},\mathbf{Dec}}(1^k, pk)$
return $b = b'$

Oracle $\mathbf{LR}(H, M_0, M_1)$:
if $|M_0| \neq |M_1|$ then
  return $\lightning$
$C \leftarrow\$ \mathsf{Enc}^H_{pk}(M_b)$
$Q \leftarrow Q \cup \{(H, C)\}$
return $C$

Oracle $\mathbf{Dec}(H, C)$:
if $(H, C) \in Q$ then
  return $\lightning$
return $\mathsf{Dec}^H_{sk}(C)$
```

```
$\mathbf{Exp}^{\text{mmr-cca}}_{\mathsf{PKEAD}}(\mathcal{A}, k)$:
$Q \leftarrow \emptyset$; $\mathsf{pkout} \leftarrow \mathsf{false}$
$(pk, sk) \leftarrow\$ \mathsf{Kgen}(1^k)$
$b \leftarrow\$ \{0, 1\}$
$b' \leftarrow\$ \mathcal{A}^{\mathbf{LR},\mathbf{Dec},\mathbf{PKout}}(1^k)$
return $b = b'$

Oracle $\mathbf{LR}(\boldsymbol{H}, \mathcal{M})$:
if $\mathsf{pkout} = \mathsf{true}$ then return $\lightning$
$(\boldsymbol{M}_0, \boldsymbol{M}_1, \boldsymbol{r}) \leftarrow\$ \mathcal{M}(1^k)$
$\boldsymbol{C} \leftarrow \mathsf{Enc}^{\boldsymbol{H}}_{pk}(\boldsymbol{M}_b\,; \boldsymbol{r})$
for $i \leftarrow 1$ to $|\boldsymbol{H}|$ do
  $Q \leftarrow Q \cup \{(\boldsymbol{H}_i, \boldsymbol{C}_i)\}$
return $\boldsymbol{C}$

Oracle $\mathbf{Dec}(H, C)$:
if $(H, C) \in Q$ then
  return $\lightning$
return $\mathsf{Dec}^H_{sk}(C)$

Oracle $\mathbf{PKout}()$:
$\mathsf{pkout} \leftarrow \mathsf{true}$; return $pk$
```

```
$\mathbf{Exp}^{\text{mm-cca}}_{\mathsf{PKEAD},\mathcal{R}}(\mathcal{A}, k)$:
$Q \leftarrow \emptyset$
$(pk, sk) \leftarrow\$ \mathsf{Kgen}(1^k)$
$b \leftarrow\$ \{0, 1\}$
$b' \leftarrow\$ \mathcal{A}^{\mathbf{LR},\mathbf{Dec}}(1^k, pk)$
return $b = b'$

Oracle $\mathbf{LR}(\boldsymbol{H}, \mathcal{M})$:
$\boldsymbol{r} \leftarrow\$ \mathcal{R}(1^k)$
$(\boldsymbol{M}_0, \boldsymbol{M}_1) \leftarrow\$ \mathcal{M}(1^k)$
$\boldsymbol{C} \leftarrow \mathsf{Enc}^{\boldsymbol{H}}_{pk}(\boldsymbol{M}_b\,; \boldsymbol{r})$
for $i \leftarrow 1$ to $|\boldsymbol{H}|$ do
  $Q \leftarrow Q \cup \{(\boldsymbol{H}_i, \boldsymbol{C}_i)\}$
return $\boldsymbol{C}$

Oracle $\mathbf{Dec}(H, C)$:
if $(H, C) \in Q$ then
  return $\lightning$
return $\mathsf{Dec}^H_{sk}(C)$
```

Fig. 2. Security notions for public-key encryption with associated data.

The corresponding notion in the chosen-plaintext attack setting is obtained by denying $\mathcal{A}$ access to **Dec**. Let $\mathbf{Exp}^{\text{mmr-cpa}}_{\mathsf{PKEAD}}(\mathcal{A}, k)$ denote this experiment and let MMR-CPA security be defined analogously to MMR-CCA.

Remarks about MMR. Notice that the adversary is not given the public key until after it is done seeing the challenge ciphertexts. It has previously been observed (in [BBN+09], building on [BBO07]) that otherwise, the adversary may craft an mmr-source, which depends on the public key, and completely leaks the challenge bit with one query. Therefore, giving the adversary the public key would render the notion unachievable.

We recall that PRIV security for deterministic encryption formalized by Bellare et al. [BBO07] is equivalent to a *non-adaptive* version of MMR-CPA (this was observed in [BBN+09]), although they do not consider PKE with AD. Bellare et al. [BBO07] also define PRIV-CCA, but it is *not* equivalent to a non-adaptive version of MMR-CCA. This is because we allow the adversary to access the decryption oracle before it learns the public key, while this is prohibited in PRIV-CCA.

Min-entropy requirements. Just as in prior work [BBO07,BBN+09], we require that the joint message-coins distribution have high min-entropy. In the MMR setting, this means the sources queried by the adversary have min-entropy $\mu = \mu(k) \in \omega(\log k)$. This is sufficient to thwart trial-encryption attacks by which the adversary, given the public key, exhaustively encrypts message-coins pairs until a ciphertext matches the output of its **LR** oracle.

```
Exp^{anon-cca}_{PKEAD}(D, k)
  $Q \gets \emptyset$
  $d \twoheadleftarrow \{0,1\}$
  $(pk_0, sk_0) \twoheadleftarrow \mathsf{Kgen}(1^k)$
  $(pk_1, sk_1) \twoheadleftarrow \mathsf{Kgen}(1^k)$
  $d' \twoheadleftarrow \mathcal{D}^{\mathbf{LR},\mathbf{Enc},\mathbf{Dec}}(1^k)$
  return $d = d'$

Oracle LR(H, M):
  if $\mathsf{pkout} = \mathsf{true}$ then return $\lightning$
  $\mathsf{pkout} \gets \mathsf{true}$
  if $(\boldsymbol{H}, \mathcal{M}) = (\bot, \bot)$ then
    return $(pk_0, pk_1, \Lambda)$
  $(\boldsymbol{M}, \boldsymbol{r}) \twoheadleftarrow \mathcal{M}(1^k)$
  $\boldsymbol{C} \gets \mathsf{Enc}(pk_d, \boldsymbol{H}, \boldsymbol{M}; \boldsymbol{r})$
  for $i \gets 1$ to $|\boldsymbol{H}|$ do
    $Q \gets Q \cup \{(\boldsymbol{H}_i, \boldsymbol{C}_i)\}$
  return $(pk_0, pk_1, \boldsymbol{C})$

Oracle Enc(H, M):
  if $\mathsf{pkout} = \mathsf{true}$ then return $\lightning$
  $(\boldsymbol{M}, \boldsymbol{r}) \twoheadleftarrow \mathcal{M}(1^k)$
  $\boldsymbol{C} \gets \mathsf{Enc}(pk_0, \boldsymbol{H}, \boldsymbol{M}; \boldsymbol{r})$
  return $\boldsymbol{C}$

Oracle Dec_b(H, C):
  if $(H, C) \in Q$ then return $\lightning$
  return $\mathsf{Dec}(sk_b, H, C)$
```

Fig. 3. Key anonymity of public-key encryption as formalized by [BBN+09], lifted to the CCA setting.

4.3 ANON security

Bellare et al. [BBN+09] studied how key anonymity is important for achieving adaptivity against MMR attacks. Unlike with the standard IND-CPA or -CCA notions, non-adaptive MMR (MMR1) security does not imply adaptive security. This is due to the fact that the adversary is not given the public key when it makes the queries to see the challenge ciphertexts. They observed that in the CPA setting, a property called key anonymity suffices to gain adaptivity. We extend their notion to the CCA setting; refer to the game defined in Figure 3.

The game begins by choosing two key pairs (pk_0, sk_0) and (pk_1, sk_1) and a challenge bit d. The adversary is executed with the security parameter as input and with access to three oracles as defined in the figure. The outcome of the game is true if and only if the adversary's output is equal to d. The output of the **LR** and **Enc** oracles is well-defined when $\boldsymbol{H} \in \mathsf{AD}^{v(k)}$ and $\mathcal{M}$ is an (μ, v, ρ)-mr-source for some $\mu, v : \mathbb{N} \to \mathbb{N}$. Following the lead of [BBDP01], we provide a decryption oracle for both the primary and alternate secret key. On input (b, H, C) where $b \in \{0,1\}$, $H \in \mathsf{AD}$, and $C \in \{0,1\}^*$, oracle **Dec** decrypts (H, C) under sk_b and returns the result as long as (H, C) was never output by **LR**.

Fix functions $\mu, v : \mathbb{N} \to \mathbb{N}$ such that $\mu(k) \in \omega(\log k)$. We define a (μ, v, ρ)-mr-adversary as one whose oracle queries consist of well-defined inputs and distinct (μ, v, ρ)-mr-sources. We say that PKEAD is ANON-CCA secure with respect to distinct (μ, v, ρ)-mr-sources if the function $\mathbf{Adv}^{\text{anon-cca}}_{\mathsf{PKEAD}}(\mathcal{A}, \cdot)$ is negligible for every PT (μ, v, ρ)-mr-adversary $\mathcal{A}$. As usual, we capture ANON-CPA security by denying the adversary access to the **Dec** oracle. This is equivalent to the ANON notion of [BBN+09], which in turn lifts [BBDP01] to the hedged setting.

Non-adaptive to adaptive MMR via ANON. Intuitively, key anonymity captures the adversary's ability to discern information about the public key given adaptively-chosen encryptions under the public key and, in our setting, decryptions under the corresponding secret key. This property suffices for the following result, lifting [BBN+09, Theorem 5.2] to the CCA setting.

Theorem 1 (MMR1+ANON-CCA $\Longrightarrow$ MMR-CCA). *Let $\mu, v, \rho : \mathbb{N} \to \mathbb{N}$ be functions where $\mu(k) \in \omega(\log k)$. Let $\mathcal{A}$ be a (μ, v, ρ)-mmr-adversary who makes q queries to its* **LR** *oracle. There exists a (μ, v, ρ)-mmr-adversary $\mathcal{B}$, who makes one query to its* **LR** *oracle, and a (μ, v, ρ)-mr-adversary $\mathcal{D}$ such that*

$$\mathbf{Adv}^{\text{mmr-cca}}_{\mathsf{PKEAD}}(\mathcal{A}, k) \le q \cdot \mathbf{Adv}^{\text{mmr-cca}}_{\mathsf{PKEAD}}(\mathcal{B}, k) + 2q \cdot \mathbf{Adv}^{\text{anon-cca}}_{\mathsf{PKEAD}}(\mathcal{D}, k)\ .$$

where $\mathcal{D}$ and $\mathcal{B}$ have the same runtime as $\mathcal{A}$. Moreover, adversary $\mathcal{D}$ makes as many decryption queries as $\mathcal{A}$, $q-1$ encryption queries, and one query to **LR**, *and adversary $\mathcal{B}$ makes as many decryption queries as $\mathcal{A}$ and one query to* **LR**.

The proof is a simple extension of [BBN+09, Theorem 5.2] that takes the decryption oracle into account; see Appendix C.1 for details. The intuition is that leakage of the public key in the ciphertext is tolerable in the non-adaptive setting since the adversary may obtain the public key after making its **LR** query. In the adaptive setting, this leakage could lead to attacks based on key-dependent message-coins distributions in subsequent **LR** queries.

Remark. We note that the converse is not true: MMR-CPA does *not* imply ANON-CPA. Suppose we modify an MMR-CPA secure PKEAD scheme by appending the hash of the public key to the end of the ciphertext. Modeling the hash function as a random oracle, this construction remains MMR-CPA secure. However, it is clearly not ANON-CPA. Since the adversary is given the primary and alternate key in response to its **LR** query, it can easily check (with one random oracle query) which key was used to encrypt.

4.4 MM security

Next, we consider the practical setting in which the coins are non-adaptively corrupted. Consider the MM-CCA experiment defined in Figure 2 associated to PKEAD, adversary $\mathcal{A}$, *randomness source* $\mathcal{R}$, and security parameter k. The output of the **LR** oracle is well-defined if for every $k \in \mathbb{N}$ and some $\mu_1, \mu_2, v : \mathbb{N} \to \mathbb{N}$, it holds that $\mathcal{M}$ is a (μ_1, v)-mm-source, $\boldsymbol{H} \in \mathsf{AD}^{v(k)}$, and $\mathcal{R}$ is a (μ_2, v, ρ)-r-source. Fix functions $\mu_1, \mu_2, v : \mathbb{N} \to \mathbb{N}$ where $\mu_2(k) \in \omega(\log k)$. We call $\mathcal{A}$ a (μ_1, v)-mm-adversary if its queries are well-defined and its **LR** queries consist of distinct (μ_1, v)-mm-sources. We say that PKEAD is MM-CCA secure with respect to distinct (μ_1, v)-mm-sources and (μ_2, v, ρ)-r-sources if for every PT (μ_1, v)-mm-adversary $\mathcal{A}$ and for every PT (μ_2, v, ρ)-r-source $\mathcal{R}$, the function $\mathbf{Adv}^{\text{mm-cca}}_{\mathsf{PKEAD},\mathcal{R}}(\mathcal{A}, \cdot)$ is negligible. Again, we let $\mathbf{Exp}^{\text{mm-cpa}}_{\mathsf{PKEAD},\mathcal{R}}(\mathcal{A}, k)$ be the experiment associated to PKEAD, $\mathcal{A}$, k, and randomness source $\mathcal{R}$, which is identical to $\mathbf{Exp}^{\text{mm-cca}}_{\mathsf{PKEAD},\mathcal{R}}(\mathcal{A}, k)$, but the adversary has no **Dec** oracle. MM-CPA security is defined analogously to MM-CCA security.

Non-adaptive to Adaptive MM "for free". Unlike in the MMR attack setting, in the MM-CCA game, the adversary is given the public key. This is achievable because the coin source may not be adaptively corrupted to depend upon it. It follows that one does get adaptivity "for free" in this setting, via a standard hybrid argument.

Theorem 2 (MM1-CCA $\Longrightarrow$ MM-CCA). *Let $\mu_1, \mu_2, v : \mathbb{N} \to \mathbb{N}$ be functions where $\mu_2(k) \in \omega(\log k)$. Let $\mathcal{R}$ be a (μ_2, v, ρ)-r-source and $\mathcal{A}$ be a (μ_1, v)-mm-adversary who makes q queries to its* **LR** *oracle. There exists a (μ_1, v)-mm-adversary $\mathcal{B}$ who makes one query to its* **LR** *oracle such that*

$$\mathbf{Adv}^{\text{mm-cca}}_{\mathsf{PKEAD},\mathcal{R}}(\mathcal{A}, k) \le q \cdot \mathbf{Adv}^{\text{mm-cca}}_{\mathsf{PKEAD},\mathcal{R}}(\mathcal{B}, k)\,,$$

and $\mathcal{B}$ has the same runtime as $\mathcal{A}$, making as many decryption queries.

Min-entropy requirements. As in the MMR setting, achieving MM security demands restrictions upon the sources. Minimally, we will need to require that $\mu_1(k)+\mu_2(k) \in \omega(\log k)$, where $\mu_1(\cdot)$ is the min-entropy of the mm-sources specified by the adversary and $\mu_2(\cdot)$ is the min-entropy of the r-source parameterizing the experiment. In fact, we need a bit more. As an illustration, suppose that $\mu_1(k) \in \omega(\log k)$ and $\mu_2(k) = 0$. This means that the randomness source always outputs the same

Result	**Shown By**
MMR-CPA (resp. MM-CPA) $\not\Longrightarrow$ ANON-CPA:	CE1
ANON-CPA $\not\Longrightarrow$ MMR-CPA (resp. MM-CPA):	CE2
MM-CPA $\not\Longrightarrow$ MMR-CPA:	CE3
IND-CPA $\not\Longrightarrow$ MM-CPA (resp. MMR-CPA):	CE4
MMR1+ANON-CCA $\Longrightarrow$ MMR-CCA	Theorem 1
MM1-CCA $\Longrightarrow$ MM-CCA	Theorem 2
MMR1+ANON-CCA $\Longrightarrow$ MM1-CCA	Theorem 3
MM-CCA $\Longrightarrow$ IND-CCA where $\mu_1(k) \in O(\log k)$	Theorem 4

Fig. 4. Summary of relations. **Top:** separations using **CE1:** $\mathsf{Enc}^H_{pk}(M;\, r) = \mathcal{E}^H_{pk}(M;\, r) \,\|\, \mathsf{H}(pk)$, where $\mathcal{E}$ is {MM,MMR}-CPA and H a random oracle; **CE2:** $\mathsf{Enc}^H_{pk}(M;\, r) = M$; **CE3:** EME-OAEP (see Section 6.1); **CE4:** $\mathsf{Enc}^H_{pk}(M;\, r \,\|\, b) = \mathcal{E}^H_{pk}(M;\, r) \,\|\, (b \oplus M[1])$, where $\mathcal{E}$ is IND-CPA. We note that the corresponding CCA separations are implied by the CPA separations. **Bottom:** implications, where we note that the corresponding CPA implications are implied by the CCA implications.

sequence of coins. This allows the adversary to mount the key-dependent distribution attack identified by [BBN+09] when the adversary is given the public key. (Indeed, this kind of attack is effective whenever the randomness source has low min-entropy. See Appendix A for details.) Therefore, it is crucial in the MM setting that the entropy of the randomness source μ_2 be of order $\omega(\log k)$.

5 Relations among the notions

We summarize the min-entropy requirements of each notion as follows: IND requires uniform random coins, MMR requires that the joint distribution on messages and coins have high min-entropy, and MM requires that the coins have high min-entropy. MMR tolerates bad randomness, but only if the message has high entropy. On the other hand, MM fails if the randomness is low min-entropy. Thus, the MM setting captures systems that are pretty good at gathering entropy, but not perfect. This is a realistic scenario, as evidenced by the analysis of the entropy-gathering mechanisms in the Linux kernel in [HDWH12]. Catastrophic failures, on the other hand, such as the infamous OpenSSL bug in the Debian distribution, which resulted in the PRNG seed having only 15 bits of entropy on many systems [Mue08], or the "boot-time entropy hole" described in [HDWH12], are out of scope. With these distinctions in mind, we study the relationships between IND, MMR, and MM attack settings. Our results are summarized in Figure 4.

Relationship between MMR and MM attacks. Intuitively, the MMR attack captures a stronger setting, since the adversary can adaptively corrupt the coins. The notions are incomparable, however, since the adversary has the public key in the MM attack setting. Nevertheless, we are able to show that a scheme that is both MMR- and ANON-CCA secure is MM-CCA secure.

Theorem 3 (MMR1+ANON-CCA $\Longrightarrow$ MM1-CCA). *Let* PKEAD *be an encryption scheme with randomness length* $\rho(\cdot)$*. Let* $\mu_1, \mu_2, v : \mathbb{N} \to \mathbb{N}$ *be functions, where* $\mu_2(k) \in \omega(\log k)$*. Let* $\mathcal{R}$ *be a* (μ_2, v, ρ)*-r-source and* $\mathcal{A}$ *be a* (μ_1, v)*-mm-adversary who makes one query to its* **LR** *oracle. There exist a* $(\mu_1{+}\mu_2, v, \rho)$*-mmr-adversary* $\mathcal{B}$ *who makes one query to its* **LR** *oracle and a* $(\mu_1{+}\mu_2, v, \rho)$*-mr adversary* $\mathcal{D}$ *such that*

$$\mathbf{Adv}^{\text{mm-cca}}_{\mathsf{PKEAD},\mathcal{R}}(\mathcal{A}, k) \le \mathbf{Adv}^{\text{mmr-cca}}_{\mathsf{PKEAD}}(\mathcal{B}, k) + 4 \cdot \mathbf{Adv}^{\text{anon-cca}}_{\mathsf{PKEAD}}(\mathcal{D}, k),$$

where and $\mathcal{B}$ and $\mathcal{D}$ have the same runtime as $\mathcal{A}$. Each makes as many decryption queries as $\mathcal{A}$ and one query to its **LR** *oracle.*

We prove this claim in Appendix C.2. Roughly speaking, our argument is that if the scheme is key anonymous, then the public key provides the adversary with negligible advantage in the MM-CCA setting. Therefore, we can give the adversary a public key different from the one used to answer its queries with it being none the wiser.

Finally, we exhibit a scheme that is MM-CPA, but *not* MMR-CPA in Section 6.1, thus concluding that MMR+ANON-CCA is a properly stronger notion than MM-CCA.

Relationship between MM and IND attacks. Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme. Define PKEAD as $(\mathcal{K}, \mathsf{Enc}, \mathsf{Dec})$ where $\mathsf{Enc}^H_{pk}(M\,;r\,\|\,b) = \mathcal{E}^H_{pk}(M\,;r)\,\|\,(b \oplus M[1])$ and $\mathsf{Dec}^H_{sk}(C\,\|\,z) = \mathcal{D}^H_{sk}(C)$. (Note that if Π has randomness length $\rho(\cdot)$, then PKEAD has randomness length $\rho(k)+1$ for all k.) Then PKEAD is IND-CPA secure as long as Π is. But PKEAD is not MM-CPA secure, since bit b might be fixed by the randomness source. It follows that IND-CPA security does not imply MM-CPA security in general. (A similar argument holds for MMR-CPA.) But what about the converse?

Recall that our notions are parameterized by the min-entropy and output length of the source(s). We may also consider finer-grained notions of security. Let $\Pi^{\text{mmr-cca}}_{\mu,v}$ denote the set of PKE schemes MMR-CCA secure with respect to distinct (μ, v, ρ)-mmr-sources, where $\rho(\cdot)$ is the randomness length of the scheme. Similarly, let $\Pi^{\text{mm-cca}}_{\mu_1,\mu_2,v}$ denote the set of PKE schemes MM-CCA secure with respect to distinct (μ_1, v)-mm-sources and (μ_2, v, ρ)-r-sources. Finally, let $\Pi^{\text{ind-cca}}$ denote the set of IND-CCA secure schemes. First, we observe that if $\varphi, \psi, v : \mathbb{N} \to \mathbb{N}$ are functions and $\varphi(k) \in O(\psi(k))$, then $\Pi^{\text{mmr-cca}}_{\varphi,v} \subseteq \Pi^{\text{mmr-cca}}_{\psi,v}$. This means that if a scheme is secure with respect to the lowest min-entropy requirement (of order $\omega(\log k)$), then it is also secure with respect to sources with more entropy. Analogously, we have that $\Pi^{\text{mm-cca}}_{\varphi_1,\varphi_2,v} \subseteq \Pi^{\text{mm-cca}}_{\psi_1,\psi_2,v}$ where $\varphi_1, \varphi_2, \psi_1, \psi_2, v : \mathbb{N} \to \mathbb{N}$ are functions such that $\varphi_1(k) \in O(\psi_1(k))$ and $\varphi_2(k) \in O(\psi_2(k))$.

As a special case, we have that $\Pi^{\text{mm-cca}}_{0,\varphi,1} \subseteq \Pi^{\text{ind-cca}}$ for every $\varphi(k) \in \omega(\log k)$. More generally, we can show that for certain classes of functions $\mu_1, \mu_2, v : \mathbb{N} \to \mathbb{N}$, it holds that $\Pi^{\text{mm-cca}}_{\mu_1,\mu_2,v} \subseteq \Pi^{\text{ind-cca}}$. First, we observe the following:

Lemma 1. *Let* PKEAD *be an encryption scheme with randomness length* $\rho(\cdot)$. *Let* $\mu_1, v : \mathbb{N} \to \mathbb{N}$ *be functions. Let* $\mathcal{A}$ *be an adversary who makes one query to its* **LR** *oracle, and* $\mathcal{U}$ *be the* (ρ, v, ρ)*-r-source defined by:* $\boldsymbol{r} \leftarrow\!\!\$\, (\{0,1\}^{\rho(k)})^{v(k)}$*; return* $\boldsymbol{r}$. *There exists a* (μ_1, v)*-mm-adversary* $\mathcal{B}$ *who makes one query to its* **LR** *oracle such that* $\mathbf{Adv}^{\text{ind-cca}}_{\mathsf{PKEAD}}(\mathcal{A}, k) \le v(k)2^{\mu_1(k)} \cdot \mathbf{Adv}^{\text{mm-cca}}_{\mathsf{PKEAD},\mathcal{U}}(\mathcal{B}, k)$*, where* $\mathsf{time}_{\mathcal{B}}(k) = \mathsf{time}_{\mathcal{A}}(k) + O(v(k)2^{\mu_1(k)})$.

Proof. Fix $k \in \mathbb{N}$ and let $\mu_1 = \mu_1(k)$, $\rho = \rho(k)$, and $v = v(k)$. Assume that $\mathcal{A}$'s query to its **LR** oracle is (H, M_0, M_1) where $H \in \mathsf{AD}$ and M_0 and M_1 are distinct, equal-length strings. This is without loss of generality, since otherwise **LR** would reject. Let $n = |M_0| = |M_1|$. We construct adversary $\mathcal{B}$ from $\mathcal{A}$. On input $(1^k, pk)$ and with oracles **LR** and **Dec**, adversary $\mathcal{B}$ executes $b' \leftarrow\!\!\$\, \mathcal{A}^{\mathbf{LR}',\mathbf{Dec}}(1^k, pk)$ and returns b', where $\mathbf{LR}'$ is defined below.

Let $\mathcal{M}$ be the following mm-source: on input 1^k, first construct a set $S \subseteq (\{0,1\}^n)^2$ such that: (1) $|S| = v2^{\mu_1}$; (2) $(M_0, M_1) \in S$; and (3) for every distinct (X_0, X_1) and (Y_0, Y_1) in S, it holds that $X_0 \ne Y_0$ and $X_1 \ne Y_1$. Next, for each $i \in [v]$, sample a pair (X, Y) uniformly and *without replacement* from S, and let $\boldsymbol{M}_0[i] = X$ and $\boldsymbol{M}_1[i] = Y$. Finally, output $(\boldsymbol{M}_0, \boldsymbol{M}_1)$. Sampling each $(\boldsymbol{M}_0[i], \boldsymbol{M}_1[i])$ without replacement means $\mathcal{M}$ is distinct. Since $|S| = v2^{\mu_1}$, for each $X \in \{0,1\}^n$,

$b \in \{0,1\}$, and $i \in [v]$, it holds that

$$\Pr\left[\,(\boldsymbol{M}_0, \boldsymbol{M}_1) \leftarrow\!\$\; \mathcal{M}(1^k) : \boldsymbol{M}_b[i] = X\,\right] \leq 1 - \frac{v2^{\mu_1}-1}{v2^{\mu_1}} \cdot \frac{v2^{\mu_1}-2}{v2^{\mu_1}-1} \cdots$$
$$= \frac{v}{v2^{\mu_1}} = \frac{1}{2^{\mu_1}}\ .$$

It follows that $\mathcal{M}$ is a distinct (μ_1, v)-mm-source. Returning now to answering $\mathcal{A}$'s **LR** queries: on input (H, M_0, M_1), oracle $\mathbf{LR}'$ first lets $\boldsymbol{H}[i] = H$ for each $i \in [v]$. It then executes $\boldsymbol{C} \leftarrow\!\$\; \mathbf{LR}(\boldsymbol{H}, \mathcal{M})$, samples $j \leftarrow\!\$\; [v]$, and returns $\boldsymbol{C}[j]$ to $\mathcal{A}$.

Adversary $\mathcal{B}$'s simulation of $\mathcal{A}$'s **LR** query (and subsequent **Dec** queries) is perfect as long as $\boldsymbol{M}_0[j] = M_0$ and $\boldsymbol{M}_1[j] = M_1$. Let good denote this event. This occurs with probability $1/v2^{\mu_1}$. Then

$$\begin{aligned}\Pr\left[\,\mathbf{Exp}^{\text{mm-cca}}_{\mathsf{PKEAD},\mathcal{U}}(\mathcal{B},k)\,\right] &= \Pr\left[\,\mathbf{Exp}^{\text{mm-cca}}_{\mathsf{PKEAD},\mathcal{U}}(\mathcal{B},k) \mid \mathsf{good}\,\right]\Pr\left[\,\mathsf{good}\,\right] + \\ &\quad + \Pr\left[\,\mathbf{Exp}^{\text{mm-cca}}_{\mathsf{PKEAD},\mathcal{U}}(\mathcal{B},k) \mid \overline{\mathsf{good}}\,\right]\Pr\left[\,\overline{\mathsf{good}}\,\right] \\ &\geq \frac{1}{v2^{\mu_1}} \cdot \Pr\left[\,\mathbf{Exp}^{\text{ind-cca}}_{\mathsf{PKEAD}}(\mathcal{A},k)\,\right]\ ,\end{aligned}$$

which yields the bound. To complete the proof, we need only to comment on the runtime of $\mathcal{B}$. Constructing the set S requires time $O(v2^{\mu_1})$. Since this dominates the time to simulate $\mathcal{A}$'s **LR** query, it follows that the runtime $\mathcal{B}$ is $\mathsf{time}_{\mathcal{A}}(k) + O(v2^{\mu_1})$. $\square$

This yields, almost immediately, the following corollary:

Theorem 4. *Let $\mu_1, \mu_2, v : \mathbb{N} \to \mathbb{N}$ be functions such that $\mu_1(k) \in O(\log k)$, $\mu_2(k) \in \omega(\log k)$, and $v(k)$ is polynomial in k. Then $\Pi^{\text{mm-cca}}_{\mu_1,\mu_2,v} \subseteq \Pi^{\text{ind-cca}}$.*

Proof. Let $\mathsf{PKEAD} \in \Pi^{\text{mm-cca}}_{\mu_1,\mu_2,v}$ have randomness length $\rho(\cdot)$. By definition, we have that $\mathsf{PKEAD} \in \Pi^{\text{mm-cca}}_{\mu_1,\rho,v}$. By Lemma 1, for every PT adversary $\mathcal{A}$, there is a PT (μ_1, v)-mm-adversary $\mathcal{B}$ such that

$$\mathbf{Adv}^{\text{ind-cca}}_{\mathsf{PKEAD}}(\mathcal{A},k) \leq v(k)2^{\mu_1(k)} \cdot \mathbf{Adv}^{\text{mm-cca}}_{\mathsf{PKEAD},\mathcal{U}}(\mathcal{B},k).$$

Hence, $\mathsf{PKEAD} \in \Pi^{\text{ind-cca}}$. $\square$

6 Constructions

In this section we present several constructions of hedged PKEAD schemes. To begin, we give a result showing that EME-OAEP (the version of RSA-OAEP that is implemented in OpenSSL) is not MMR-CPA, but is provably MM-CCA in the ROM, under a standard assumption on RSA. This gives the first positive result for RSA-OAEP in the presence of imperfect randomness, and is callable via the high-level APIs exposed by all major libraries.

To achieve MMR+IND-CCA, we give a hybrid-encryption PKEAD scheme. This, too, can be realized by high-level API calls in modern libraries, using RSA as the trapdoor function, and available hash function and symmetric authenticated encryption functionalities.

We then revisit the generic compositions RtD and PtD from Bellare et al. [BBN+09]. We show that if the deterministic scheme is instantiated specifically by RSA-DOAEP [BBO07], which can be done via high-level API calls to hash functions and RSA, then PtD achieves MM+IND-CCA,

```
$\underline{\mathsf{Kgen}(1^k)}$
$(f, f^{-1}) \leftarrow\!\!\$\; F(1^k)$
return $(\langle f\rangle, \langle f^{-1}\rangle)$

$\underline{\mathsf{Enc}^H_{pk}(M)}$
$\langle f\rangle \leftarrow pk$; $PM \leftarrow \mathsf{pad}(M)$
if $PM = \perp$ then return $\perp$
$X_0 \leftarrow PM \,\|\, \mathsf{H}_1(H)$
$Y_0 \leftarrow\!\!\$\; \{0,1\}^\rho$
$X_1 \leftarrow X_0 \oplus \mathsf{G}(Y_0)$
$Y_1 \leftarrow Y_0 \oplus \mathsf{H}_2(X_1)$
$P \leftarrow X_1 \,\|\, Y_1 \,\|\, [0]$
return $f(P)$

$\underline{\mathsf{Dec}^H_{sk}(C)}$
$\langle f^{-1}\rangle \leftarrow sk$; $P \leftarrow f^{-1}(C)$
if $|P| \neq n$ then return $\perp$
$X_1 \,\|\, Y_1 \,\|\, [z] \leftarrow P$   # $|Y_1| = \rho$
$Y_0 \leftarrow Y_1 \oplus \mathsf{H}_2(X_1)$
$X_0 \leftarrow X_1 \oplus \mathsf{G}(Y_0)$
$PM \,\|\, T \leftarrow X_0$   # $|T| = \tau$
if $\mathsf{H}_1(H) \neq T$ then return $\perp$
return $\mathsf{unpad}(PM)$
```

Fig. 5. Specification of F-EME-OAEP encryption (RFC 8017) where F is a trapdoor permutation generator with input length $n(\cdot)$. Let $\tau(\cdot)$ and $\rho(\cdot)$ be functions where for every $k \in \mathbb{N}$, it holds that $\rho(k)+\tau(k)+16 \leq n(k)$. Fix $k \in \mathbb{N}$ and let $n = n(k)$, $\tau = \tau(k)$, $\rho = \rho(k)$, and $m = n - \rho - 8$. The syntax $[i]$ denotes integer i, where $0 \leq i \leq 255$, encoded as a byte. Let $\mathsf{H}_1 : \{0,1\}^* \to \{0,1\}^\tau$, $\mathsf{G} : \{0,1\}^* \to \{0,1\}^m$, and $\mathsf{H}_2 : \{0,1\}^* \to \{0,1\}^\rho$ be functions. Define $\mathsf{pad} : \{0,1\}^* \to \{0,1\}^{m-\tau} \cup \{\perp\}$ by $\mathsf{pad}(M) = M \,\|\, [1] \,\|\, [0] \cdots [0]$ if $|M|$ is less than or equal to $m - \tau - 8$ and is a multiple of 8, and $\mathsf{pad}(M) = \perp$ otherwise. Define its inverse $\mathsf{unpad} : \{0,1\}^{m-\tau} \to \{0,1\}^* \cup \{\perp\}$ in the natural way.

and RtD achieves MM+IND-CPA. We also suggest specific conditions under which RtD would be MMR+IND-CCA, extending prior work [BBN+09].

Trapdoor permutations. Some of our constructions make use of trapdoor permutations, so we recall this primitive and its security here. Let $k \in \mathbb{N}$. A *trapdoor permutation generator* is a probabilistic algorithm F with associated input length[7] $n(\cdot)$ that on input 1^k outputs the encoding of a pair of functions $f, f^{-1} : \{0,1\}^* \to \{0,1\}^*$ such that for every $x \in \{0,1\}^{n(k)}$, it holds that $f^{-1}(f(x)) = x$. We say that F is OWF secure if for every PT adversary $\mathcal{A}$, the quantity

$$\mathbf{Adv}^{\mathrm{owf}}_{F}(\mathcal{A}, k) = \Pr\Big[(f, f^{-1}) \leftarrow\!\!\$\; F(1^k); x \leftarrow\!\!\$\; \{0,1\}^{n(k)} : \mathcal{A}(1^k, f, f(x)) \Rightarrow x\Big]$$

is a negligible function of k.

We will also use the stronger security notion of *partial-domain one-wayness* formalized by Fujisaki et al. [FOPS04], which asserts that it is difficult to partially invert a value in the range of the trapdoor permutation. Let F be a trapdoor permutation generator with input length $n(\cdot)$ and let $m(\cdot)$ be a function such that $m(k) \leq n(k)$ for every $k \in \mathbb{N}$. We say that F is m-POWF secure if for every PT adversary $\mathcal{A}$, the following function is negligible in k:

$$\mathbf{Adv}^{\mathrm{powf}}_{F,m}(\mathcal{A}, k) = \Pr\Big[(f, f^{-1}) \leftarrow\!\!\$\; F(1^k); x \leftarrow\!\!\$\; \{0,1\}^{n(k)} : \mathcal{A}(1^k, f, f(x)) \Rightarrow x[1..m(k)]\Big] .$$

6.1 EME-OAEP

We first look at RSA-OAEP [BR95], the only provably-secure PKE scheme available in OpenSSL, and indeed most libraries.[8] It is known to be IND-CCA secure assuming that the underlying trapdoor permutation is POWF secure, or under the RSA assumption [Sho02, FOPS04].

[7] For example, the input length might be the number of modulus bits in RSA.

[8] Some implement ElGamal or hybrid encryption schemes as well.

We specify the EME-OAEP variant standardized in PKCS #1 version 2.2 (RFC 8017). Let F be a trapdoor permutation generator. Refer to the encryption scheme F-$\mathsf{EME\text{-}OAEP}$ specified in Figure 5. This scheme resembles standard OAEP except that a hash of the associated data (called a *label* in RFC 8017) is appended to the message.[9] Instead of checking for a string of zero-bytes, the decrypting party checks that the hash of the associated data matches. In addition, a zero-byte is appended to the pad before applying the trapdoor.[10]

F-$\mathsf{EME\text{-}OAEP}$ IS NOT MMR-CPA. This scheme is not MMR-CPA secure, due to an attack by Brown [Bro05] on $\mathsf{RSA\text{-}OAEP}$ with exponent $e = 3$. The attack exploits low entropy coins. An adversary who knows (or is able to guess) the coins can recover the entire plaintext, meaning the attack is effective even if the message has high min-entropy. Since this attack does not exploit the tag used to check if the ciphertext is valid during decryption, it is equally effective in breaking $\mathsf{RSA\text{-}EME\text{-}OAEP}$.

F-$\mathsf{EME\text{-}OAEP}$ IS MM-CCA. We prove the scheme does achieve our new notion. The standard cites the result of [FOPS04] to establish the IND-CCA security of this scheme, but this result makes no formal claim for the security of the associated data. Moreover, no security guarantee is known in case randomness is not perfect. We extend their analysis to account for associated data and imperfect randomness and prove, in the random oracle model, that F-$\mathsf{EME\text{-}OAEP}$ is MM-CCA secure with respect to high min-entropy coins sources, assuming that F is POWF secure. By [FOPS04, Lemma 4.2], instantiating the trapdoor with RSA is secure assuming only that RSA is OWF secure.

Theorem 5 (F-$\mathsf{EME\text{-}OAEP}$ is MM-CCA). *Let F be a length $n(\cdot)$ trapdoor permutation generator. Let $\mu_1, \mu_2, v, \tau, \rho : \mathbb{N} \to \mathbb{N}$ be functions where $\mu_2(k) \in \omega(\log k)$ and $\rho(k) + \tau(k) + 16 \le n(k)$ for every $k \in \mathbb{N}$. Let $m(k) = n(k) - \rho(k) - 8$. Let $\mathsf{PKEAD} = F$-$\mathsf{EME\text{-}OAEP}$ as defined in Figure 5, where H_1, H_2, and G are modeled as random oracles. Let $\mathcal{A}$ be a (μ_1, v)-mm-adversary who makes q_e queries to* **LR**, *q_d queries to* **Dec**, *and q_1, q_2, and q_G queries to H_1, H_2, and G respectively. Let $\mathcal{R}$ be a (μ_2, v, ρ)-r-source. There exists an adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\text{mm-cca}}_{\mathsf{PKEAD},\mathcal{R}}(\mathcal{A}, k) \le 512 q_e q_2 v(k) \cdot \mathbf{Adv}^{\text{powf}}_{F,m}(\mathcal{B}, k) + \frac{q_e(q_1 + q_d)^2}{2^{\tau(k)-1}} + \frac{q_e(q_G + q_d)^2}{2^{\rho(k)-1}} + \frac{q_e v(k)(q_G + q_d)}{2^{\mu_2(k)-1}} ,$$

where $\mathsf{time}_{\mathcal{B}}(k) = \mathsf{time}_{\mathcal{A}}(k) + O(q_d q_1 q_G q_2)$.

Refer to Appendix C.3 for the full proof. Note that the security bound does not depend on the min-entropy of the message source, but only on the min-entropy of the randomness source. This is undesirable from a concrete security standpoint, since any entropy in the messages is thrown away. In Section 6.3, we show that adding an additional Feistel round is sufficient to establish a concrete security bound that depends on the message entropy. Note that the loss of 2^8 in the bound is the result of fixing the most significant byte as $[0]$.

In real-world terms, this result suggests that it is safe to use $\mathsf{RSA\text{-}EME\text{-}OAEP}$ barring catastrophic failure of the (P)RNG. If the adversary is able to guess the coins used, then there is an attack [Bro05], and so the Dual EC DRBG attack [CMG+16], for example, completely breaks the security of $\mathsf{RSA\text{-}EME\text{-}OAEP}$. Even cases where the coins still have *some* entropy [Mue08] we consider insecure in an asymptotic sense, since an adversary can guess the coins with non-negligible probability.

[9] Interestingly, no API we surveyed exposes AD as a parameter, although the standard supports AD.

[10] The zero-byte is intended to ensure that the message is in $\mathbb{Z}_N^*$ in the case of RSA.

MMR DOES NOT IMPLY MM SECURITY. Since F-$\mathsf{EME\text{-}OAEP}$ is not MMR-CPA, we conclude that MMR-CPA does not imply MM-CPA in general.

6.2 Hybrid encryption construction

Next, we present a novel scheme that is MMR-CCA in the random oracle model, and at the same time can be implemented using most high-level APIs, including OpenSSL. The scheme is a hybrid construction combining a trapdoor permutation, an authenticated encryption scheme with authenticated data (AEAD, now a standard notion in crypto libraries), and hash functions modeled as random oracles. We recall the notion of AEAD and then proceed to define the PKEAD scheme.

AUTHENTICATED ENCRYPTION WITH ASSOCIATED DATA (AEAD). An AEAD scheme consists of three algorithms $\mathsf{AEAD} = (\mathsf{Kgen}, \mathsf{Enc}, \mathsf{Dec})$. The randomized *key generation* algorithm Kgen samples a key K from a finite, non-empty set $\mathcal{K}$ called the *key space*. The deterministic *encryption algorithm* $\mathsf{Enc}\colon \mathcal{K} \times \mathcal{N} \times \mathsf{AD} \times \{0,1\}^* \to \{0,1\}^* \cup \{\perp\}$ takes as input a key K, a nonce $N \in \mathcal{N}$, associated data $H \in \mathsf{AD}$, and a message $M \in \{0,1\}^*$, and it returns a ciphertext $C \in \{0,1\}^*$ or the distinguished symbol $\perp$. We sometimes write $C \leftarrow \mathsf{Enc}_K^{H,N}(M)$ as a shorthand for $C \leftarrow \mathsf{Enc}(K, N, H, M)$. The deterministic *decryption algorithm* $\mathsf{Dec}\colon \mathcal{K} \times \mathcal{N} \times \mathsf{AD} \times \{0,1\}^* \to \{0,1\}^* \cup \{\perp\}$ takes as input a key K, a nonce $N \in \mathcal{N}$, associated data $H \in \mathsf{AD}$, and ciphertext $C \in \{0,1\}^*$, and outputs either the plaintext M or $\perp$. We sometimes write $M \leftarrow \mathsf{Dec}_K^{H,N}(C)$ as shorthand for $M \leftarrow \mathsf{Dec}(K, N, H, C)$. For correctness, it is required that for all $K \in \mathcal{K}$, $H \in \mathsf{AD}$, $N \in \mathcal{N}$ and $M \in \{0,1\}^*$, we have $\mathsf{Enc}_K^{H,N}(M) \neq \perp \implies \mathsf{Dec}_K^{H,N}(\mathsf{Enc}_K^{H,N}(M)) = M$.

MESSAGE PRIVACY. To define message privacy, let $\mathcal{A}$ be an adversary and consider the experiment $\mathbf{Exp}_{\mathsf{AEAD}}^{\text{ind-cpa}}(\mathcal{A})$. The experiment first generates the key $K \leftarrow_{\$} \mathsf{Kgen}$ and samples a bit $b \leftarrow_{\$} \{0,1\}$. The adversary has access to the encryption oracle $\mathsf{Enc}(K, \cdot, \cdot, LR(\cdot, \cdot, b))$, where $LR(\cdot, \cdot, b)$ on inputs $M_0, M_1 \in \{0,1\}^*$ with $|M_0| = |M_1|$ returns M_b. We say that $\mathcal{A}$ is *nonce-respecting* if it never repeats N in its oracle queries. (Hereafter, we assume the IND-CPA attacker is nonce-respecting.) Finally, adversary $\mathcal{A}$ outputs a bit b'. The outcome of the game is the predicate $(b = b')$. We define $\mathcal{A}$'s advantage as $\mathbf{Adv}_{\mathsf{AEAD}}^{\text{ind-cpa}}(\mathcal{A}) = 2 \cdot \Pr[\mathbf{Exp}_{\mathsf{AEAD}}^{\text{ind-cpa}}(\mathcal{A})] - 1$.

AUTHENTICITY. To define message authenticity, let $\mathcal{A}$ be an adversary and consider the experiment $\mathbf{Exp}_{\mathsf{AEAD}}^{\text{auth}}(\mathcal{A})$. It first generates a key $K \leftarrow_{\$} \mathsf{Kgen}$, then provides $\mathcal{A}$ access to oracle $\mathsf{Enc}(K, \cdot, \cdot, \cdot)$. (Note that the AUTH adversary need not be nonce-respecting.) The adversary can also query a special decryption oracle on triples (N, H, C). This oracle returns 1 if $\mathsf{Dec}_K^{H,IV}(C) \neq \perp$, and 0 otherwise. The game outputs true if and only if the special decryption oracle returns 1 on some query (N, H, C) and $\mathcal{A}$ never queried (N, H, M) for some $M \in \{0,1\}^*$ and got C in response. Let $\mathbf{Adv}_{\mathsf{AEAD}}^{\text{auth}}(\mathcal{A}) = \Pr[\mathbf{Exp}_{\mathsf{AEAD}}^{\text{auth}}(\mathcal{A})]$.

HYBRID PKEAD FROM A TDP AND AEAD. We propose a PKEAD scheme that uses a trapdoor permutation and an AEAD symmetric encryption scheme. Its algorithms can be implemented using the library calls to RSA function with no padding and to any AEAD scheme such as AES-GCM. The scheme is defined in Figure 6. The functions H_1 and H_2 are realized using cryptographic hash functions, but are modeled as random oracles in the analysis. We assume that there is an efficient function $\mathsf{extract}$ that on input associated data $\tilde{H}$ returns the n-bit nonce for AEAD scheme. The goal of $\mathsf{extract}$ is to make sure that the outputs do not repeat. If H contains a counter, or some other non-repeating string, then that could be used as an extracted nonce. Alternatively, C_1 or its part could be used as a nonce. (In the analysis we take into account that the asymmetric parts of

$\underline{\mathsf{Kgen}(1^k)}$:	$\underline{\mathsf{Enc}_{pk}(M,H)}$	$\underline{\mathsf{Dec}_{sk}(H, C_1 \,\|\, C_2)}$
$(f, f^{-1}) \leftarrow\!\!{\scriptstyle\$}\; F(1^k)$ $R \leftarrow\!\!{\scriptstyle\$}\; \{0,1\}^r$ return $(f \,\|\, R, f^{-1})$	$X \leftarrow\!\!{\scriptstyle\$}\; \{0,1\}^{\rho(k)}$ $K_P \leftarrow \mathsf{H}_1(\langle f \,\|\, R, H, M, X\rangle)$ $C_1 \leftarrow f(K_P)$ $K \leftarrow \mathsf{H}_2(\langle f \,\|\, R, H, K_P\rangle)$ $\tilde{H} \leftarrow \langle H, C_1\rangle$ $N \leftarrow \mathsf{extract}(\tilde{H})$ $C_2 \leftarrow \mathsf{AEAD.Enc}(K, N, \tilde{H}, M)$ return $C_1 \,\|\, C_2$	$K_P \leftarrow f^{-1}(C_1)$ $K \leftarrow \mathsf{H}_2(\langle f \,\|\, R, H, K_P\rangle)$ $\tilde{H} \leftarrow \langle H, C_1\rangle$ $N \leftarrow \mathsf{extract}(\tilde{H})$ $M \leftarrow \mathsf{AEAD.Dec}(K, N, \tilde{H}, C_2)$ return M

Fig. 6. Hybrid encryption construction $\mathsf{HE}[F, \mathsf{AEAD}]$ with randomness length $\rho(\cdot)$ and additional parameters $n, \lambda, k_P \in \mathbb{N}$. Let F be a length $n(\cdot)$ trapdoor permutation generator, such that $n(k) \geq k_P$ for sufficiently large k, and let AEAD be an AEAD scheme with key space $\{0,1\}^\lambda$, nonce space $\{0,1\}^n$, and associated-data space $\{0,1\}^*$. Let $\mathsf{H}_1 : \{0,1\}^* \to \{0,1\}^{k_P}$ and $\mathsf{H}_2 : \{0,1\}^* \to \{0,1\}^\lambda$ be functions. Let $\mathsf{extract}\colon \{0,1\}^* \to \{0,1\}^n$ be a function that on input $\tilde{H}$ returns the n-bit nonce.

ciphertexts do not repeat with overwhelming probability.) We leave the particular instantiation of $\mathsf{extract}$ to the applications.

$\mathsf{HE}[F, \mathsf{AEAD}]$ IS MMR+IND-CCA. The following theorem establishes MMR- and IND-CCA security of our hybrid construction.

Theorem 6. *Let F be a trapdoor permutation generator, AEAD be an AEAD scheme, and $\mathsf{PKEAD} = \mathsf{HE}[F, \mathsf{AEAD}]$ as defined in Figure 6, where H_1 and H_2 are modeled as random oracles.*

– *(MMR-CCA) Let $\mu, v : \mathbb{N} \to \mathbb{N}$ be functions such that $\mu(k) \in \omega(\log k)$. Let $\mathcal{A}$ be a (μ, v, ρ)-mmr-adversary attacking PKEAD and making q queries to its **LR** oracle, q_d queries to its **Dec** oracle, and q_{H_1} and q_{H_2} queries to H_1 and H_2 respectively. Then there exist adversary $\mathcal{B}$ attacking F and adversaries $\mathcal{C}$ and $\mathcal{D}$ attacking AEAD, such that*

$$\begin{aligned}\mathbf{Adv}^{\text{mmr-cca}}_{\mathsf{PKEAD}}(\mathcal{A}, k) \leq{}& \frac{q_{\mathsf{H}_1} + q_d}{2^{r-1}} + \frac{(q_{\mathsf{H}_1} + q^2 v(k))}{2^{\mu(k)-1}} + \frac{q_d + q^2 v^2(k)}{2^{k_P - 1}} \\ &+ 2v(k)q \cdot \Big(\mathbf{Adv}^{\text{owf}}_{F}(\mathcal{B}, k) + \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{AEAD}}(\mathcal{C}, k) + \mathbf{Adv}^{\text{auth}}_{\mathsf{AEAD}}(\mathcal{D}, k)\Big) .\end{aligned}$$

– *(IND-CCA) Let $\mathcal{A}$ be an adversary attacking PKEAD and making q queries to its **LR** oracle, q_d queries to its **Dec** oracle, and q_{H_1} and q_{H_2} queries to H_1 and H_2. Then there exist an adversary $\mathcal{B}$ attacking F and adversaries $\mathcal{C}$ and $\mathcal{D}$ attacking AEAD, such that*

$$\begin{aligned}\mathbf{Adv}^{\text{ind-cca}}_{\mathsf{PKEAD}}(\mathcal{A}, k) \leq{}& \frac{q_{\mathsf{H}_1}}{2^{\rho(k)-1}} + \frac{q_d}{2^{k_P - 1}} \\ &+ 2v(k)q \cdot \Big(\mathbf{Adv}^{\text{owf}}_{F}(\mathcal{B}, k) + \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{AEAD}}(\mathcal{C}, k) + \mathbf{Adv}^{\text{auth}}_{\mathsf{AEAD}}(\mathcal{D}, k)\Big) .\end{aligned}$$

In both cases, we have that $\mathsf{time}_{\mathcal{B}}(k), \mathsf{time}_{\mathcal{C}}(k), \mathsf{time}_{\mathcal{D}}(k) \approx \mathsf{time}_{\mathcal{A}}(k)$, $\mathcal{C}$ makes at most $v(k)q$ queries to its encryption oracle, and $\mathcal{D}$ makes $v(k)q$ queries to its encryption oracle, and q_d queries to its decryption oracle.

The proof is in Appendix C.4. Here we sketch the more challenging proof of MMR-CCA security. We consider a sequence of games that starts with the MMR-CCA experiment and ends with the one where random messages are encrypted with the $\mathsf{AEAD.Enc}$ under random keys, which are independent from the asymmetric ciphertexts. The view of the adversary in the last game is independent of the challenge bit. As we move between games, we consider a series of "bad" events. The first bad event happens if the H_1 oracle is queried on the values colliding with those output by the mmr-source during encryption computation. We can bound such an event by relying on the entropy of the mmr-source, if the collision occurs after the public key is revealed, or using the fact that the adversary does not know the public key and cannot guess its randomizer value if the collision happens before the public key is revealed. If this "bad" event never happens, then K_p values used to compute the asymmetric parts of the challenge ciphertexts can be chosen at random. Another bad event is set when a H_2 oracle query is made so it contains the K_p that was used as input to f during encryption. If this does not happen, we can use random symmetric keys for $\mathsf{AEAD.Enc}$. If this bad event does happen, we can construct the OWF adversary for trapdoor permutation generator F. Once we are in a game where random symmetric keys are used, we can use the IND-CPA security of AEAD. Here we have to make sure that the IND-CPA adversary is nonce-respecting. This follows from the fact that the asymmetric parts of the challenge ciphertexts, from which nonces are derived, do not repeat with overwhelming probability.

Care is needed to ensure that the adversary does not get information about the public key from the decryption queries and that the adversaries we construct can answer the decryption oracle queries. If the adversary makes a valid decryption oracle query, so that the asymmetric part is the same as that of some challenge ciphertext, then we can construct an adversary breaking authenticity of the AEAD scheme. If the asymmetric part of the ciphertext in the decryption oracle query is new, i.e., it is different from those of all challenge ciphertexts, and no corresponding H_2 query was made, the ciphertext can be rejected, as it can be valid only with negligible probability. Before the public key is revealed, such a hash query can only be made by the adversary with negligible probability. If the public key has been revealed, than such a ciphertext can be decrypted without the knowledge of the secret key.

We remark that it is not necessary to assume that the base symmetric encryption scheme provides authenticity. Instead we could rely on the random oracle model. We chose to use authenticity mostly because this makes our (already complicated) proof slightly easier.

6.3 Generic constructions

We describe two black-box constructions of [BBN+09], which compose generic randomized and *deterministic* encryption schemes. Appealing to the security properties of their constituents, these constructions are shown to be MMR+IND-CPA secure in the standard model. We consider lifting these results to the CCA setting, and consider security against MM attacks. First, we specify deterministic encryption and briefly describe its associated security notions. It will be convenient formulate the syntax without associated data.

Deterministic encryption. A deterministic PKE scheme Π is a triple of algorithms $(\mathcal{K}, \mathcal{E}, \mathcal{D})$. On input 1^k, algorithm $\mathcal{K}$ probabilistically outputs a key pair (pk, sk). Encryption deterministically maps the public key pk and a string M to an element of $\{0,1\}^* \cup \{\perp\}$. Decryption deterministically maps the secret key sk and a string C to an element of $\{0,1\}^* \cup \{\perp\}$. The scheme is correct if for every $k \in \mathbb{N}$, $(pk, sk) \in [\mathcal{K}(1^k)]$, and $M \in \{0,1\}^*$, it holds that $\mathcal{E}_{pk}(M) \neq \perp$ implies $\mathcal{D}_{sk}(\mathcal{E}_{pk}(M)) = M$. It will be helpful to assume that deterministic schemes are defined on all strings of a particular length. We say Π has input length $n(\cdot)$ if encryption is defined for all strings of length $n(k)$ and all k.

We consider both MMR-CPA and -CCA security of deterministic schemes against $(\mu, v, 0)$-mmr adversaries for functions $\mu, v : \mathbb{N} \to \mathbb{N}$, where $\mu(k) \in \omega(\log k)$. In order to instantiate a deterministic scheme in the game, we allow encryption to take coins as input, but these are simply ignored. Similarly, we allow encryption and decryption to take associated data as input, but this is ignored. Note that it does not make sense to consider MM-CPA or -CCA security of deterministic schemes, since we cannot defend against key-dependent distribution attacks in this setting. Security of deterministic encryption was first formalized by [BBO07]. Their CPA notion, PRIV, is equivalent to MMR1-CPA security. However, their CCA notion, PRIV-CCA, is *not* equivalent to MMR1-CCA. In our notion, the message source specified by the adversary is allowed to depend on prior decryption queries, whereas in the PRIV-CCA game, the adversary makes decryption queries only after it gets its challenge.

Block-sources. Recall the notion of an $\mathrm{m}^\beta\mathrm{r}^\gamma$-source given in Section 4. In the standard model, we consider security with respect to $\mathrm{m}^\beta\mathrm{r}^\gamma$*-block-sources*, where the outputs have high *conditional* min-entropy. Intuitively, this means that, from the adversary's perspective, each output of a block-source has high min-entropy even having seen the prior elements of the vector. (See [BBN+09] for a precise definition.)

Lossy and all-but-one trapdoor functions. LTDFs were first described by Peikert and Waters [PW08]. Informally, an *LTDF generator* F is a probabilistic algorithm with input length $n(\cdot)$ that on input 1^k and $b \in \{0,1\}$ outputs a pair of strings (s,t) such that s encodes a function $f : \{0,1\}^{n(k)} \to \{0,1\}^*$ with the following properties: One, if $b = 1$, then function f is injective, and t encodes a function f^{-1} giving its inverse; two, if $b = 0$, then the image of f is significantly smaller than the injective mode, i.e., $b = 1$; and three, no reasonable adversary, given s, can distinguish the injective mode from the lossy mode, i.e., $b = 0$. We call F *universal-inducing* if the lossy mode is a universal hash function.

Motivated by the goal of instantiating IND-CCA secure probabilistic encryption, [PW08] introduce ABO ("all-but-one") TDFs as a richer abstraction. An *ABO TDF generator* is a probabilistic algorithm G with an associated input length $n(\cdot)$ and a finite set B, called the *branch set*. On input 1^k and $b^* \in B$, called the *lossy* branch, it outputs (the encodings of) a pair of functions $g : B \times \{0,1\}^{n(k)} \to \{0,1\}^*$ and $g^{-1} : B\, times \{0,1\}^* \to \{0,1\}^{n(k)}$ such that: One, for every $b \in B$, if $b \neq b^*$, then $g(b, \cdot)$ is injective and $g^{-1}(b, \cdot)$ is its inverse; two; function $g(b^*, \cdot)$ is lossy; and three, for any two branches $a, b \in B$, the (bit string encodings of) functions $g(a, \cdot)$ and $g(b, \cdot)$ are computationally indistinguishable. Both primitives have been constructed from a number of hardness assumptions: For example, the Φ-hiding assumption for RSA [KOS10] and LWE ("learning with errors") for lattices [PW08]. A universal LTDF is given by Boldyreva, Fehr, and O'Neill [BFO08] based on the DDH assumption.

Pad-then-Deterministic. The transformation of a deterministic encryption scheme into a probabilistic one via a randomized padding scheme is defined in the top panel of Figure 7. This is the same as the construction proposed by [BBN+09], except we account for associated data. the message space of $\mathsf{PtD}[\Pi]$ is determined by Π. The associated data is restricted to bit strings of the length $k_0(\cdot)$. We first review the results known for PtD in the standard model, then consider its extension to the MMR- and MM-CCA settings.

Let Π be a deterministic scheme and $\mathsf{PtD}[\Pi]$ be as defined in Figure 7. Bellare et al. [BBN+09, Theorem 6.3] prove this construction is MMR-CPA if Π is MMR-CPA, and IND-CPA if Π is a

```
$\underline{\mathsf{PtD}[\Pi_d].\mathsf{Kgen}(1^k)}$
    $(pk, sk) \leftarrow\!\!\$\ \mathcal{K}_d(1^k)$
    return $(pk, sk)$
```

```
$\underline{\mathsf{PtD}[\Pi_d].\mathsf{Enc}^H_{pk}(M)}$
    if $|H| \neq k_0$ then return $\perp$
    $r \leftarrow\!\!\$\ \{0,1\}^\rho$
    $PM \leftarrow \mathsf{pad}_{n-k_0}(\langle M, r\rangle)$
    return $\mathcal{E}_d(pk, H \,\|\, PM)$
```

```
$\underline{\mathsf{PtD}[\Pi_d].\mathsf{Dec}^H_{sk}(C)}$
    $H' \,\|\, PM \leftarrow \mathcal{D}_d(sk, C)$  $\#\ |H'| = k_0$
    if $H' \neq H$ then return $\perp$
    $\langle M, r\rangle \leftarrow \mathsf{unpad}_{n-k_0}(PM)$
    return $M$
```

```
$\underline{\mathsf{RtD}[\Pi_r, \Pi_d].\mathsf{Kgen}(1^k)}$
    $(pk_r, sk_r) \leftarrow\!\!\$\ \mathcal{K}_r(1^k)$
    $(pk_d, sk_d) \leftarrow\!\!\$\ \mathcal{K}_d(1^k)$
    return $(\langle pk_r, pk_d\rangle, \langle sk_r, sk_d\rangle)$
```

```
$\underline{\mathsf{RtD}[\Pi_r, \Pi_d].\mathsf{Enc}^H_{pk}(M)}$
    $\langle pk_r, pk_d\rangle \leftarrow pk$
    $C' \leftarrow\!\!\$\ \mathcal{E}_r(pk_r, H, M)$
    return $\mathcal{E}_d(pk_d, \mathsf{pad}_n(C'))$
```

```
$\underline{\mathsf{RtD}[\Pi_r, \Pi_d].\mathsf{Dec}^H_{sk}(C)}$
    $\langle sk_r, sk_d\rangle \leftarrow sk$
    $X \leftarrow \mathcal{D}_d(sk_d, C)$
    $C' \leftarrow \mathsf{unpad}_n(X)$
    return $\mathcal{D}_r(sk_r, H, C')$
```

```
$\underline{F\text{-}\mathsf{DOAEP}.\mathcal{K}(1^k)}$
    $(f, f^{-1}) \leftarrow\!\!\$\ F(1^k)$
    return $(\langle f\rangle, \langle f^{-1}\rangle)$
```

```
$\underline{F\text{-}\mathsf{DOAEP}.\mathcal{E}_{pk}(X)}$
    if $|X| \neq n$ then return $\perp$
    $\langle f\rangle \leftarrow pk$
    $X_\ell \leftarrow X[1..k_0]$
    $X_r \leftarrow X[k_0+1..|X|]$
    $S_0 \leftarrow \mathsf{H}_1(pk \,\|\, X_r) \oplus X_\ell$
    $T_0 \leftarrow \mathsf{G}(pk \,\|\, S_0) \oplus X_r$
    $S_1 \leftarrow \mathsf{H}_2(pk \,\|\, T_0) \oplus S_0$
    $Y_\ell \,\|\, Y_r \leftarrow S_1 \,\|\, T_0$  $\#\ |Y_r| = k_1$
    return $Y_\ell \,\|\, f(Y_r)$
```

```
$\underline{F\text{-}\mathsf{DOAEP}.\mathcal{D}_{sk}(Y)}$
    if $|Y| < n - k_1$ then return $\perp$
    $\langle f^{-1}\rangle \leftarrow sk$
    $Y_\ell \leftarrow Y[1..a]$
    $Y_r \leftarrow f^{-1}(Y[a+1..|Y|])$
    $S_1 \,\|\, T_0 \leftarrow Y_\ell \,\|\, Y_r i$  $\#\ |S_1| = k_0$
    $S_0 \leftarrow \mathsf{H}_2(pk \,\|\, T_0) \oplus S_1$
    $X_r \leftarrow \mathsf{G}(pk \,\|\, S_0) \oplus T_0$
    $X_\ell \leftarrow \mathsf{H}_1(pk \,\|\, X_r) \oplus S_0$
    return $X_\ell \,\|\, X_r$
```

Fig. 7. Generic constructions. Let $k_0, k_1, n, \rho : \mathbb{N} \to \mathbb{N}$ be such that $k_0(k) + \rho(k) \leq n(k)$ for all k. Let $\Pi_d = (\mathcal{K}_d, \mathcal{E}_d, \mathcal{D}_d)$ be a deterministic scheme with input length $n(\cdot)$ and let $\Pi_r = (\mathcal{K}_r, \mathcal{E}_r, \mathcal{D}_r)$ be a randomized encryption scheme. Let F be a trapdoor permutation generator with input length $k_1(\cdot)$. Let $\mathsf{pad}_\ell : \{0,1\}^* \to \{0,1\}^\ell \cup \{\perp\}$ be an invertible encoding scheme with $\mathsf{unpad}_\ell : \{0,1\}^* \to \{0,1\}^* \cup \{\perp\}$ as its inverse. Fix $k \in \mathbb{N}$ and let $k_0 = k_0(k)$, $k_1 = k_1(k)$, $n = n(k)$, $\rho = \rho(k)$, and $a = \max\{0, n - k_1\}$. If Y is a string and $a \leq 0$, then let $Y[1..a] = \varepsilon$. Let $\mathsf{H}_1, \mathsf{H}_2 : \{0,1\}^* \to \{0,1\}^{k_0}$ and $\mathsf{G} : \{0,1\}^* \to \{0,1\}^{n-k_0}$ be functions.

u-LTDF. [11] By Theorem 1, any scheme that is both MMR1- and ANON-CPA secure is also MMR-CPA secure. If Π is a u-LTDF, then it is MMR1-CPA secure for block-sources [BFO08, Theorem 5.1], and ANON-CPA secure for block-sources [BBN+09, Theorem 5.3]. Thus, the scheme $\mathsf{PtD}[\Pi]$ is MMR-hedged secure (for block-sources) against chosen-distribution attacks as long as Π is a u-LTDF. Note that universal-inducing property is not essential; see [BBN+09, Section 6.2] for details.

Unfortunately, this property of the base scheme does not suffice for security in the CCA setting. Nevertheless, a similar construction gets us a step in the right direction. Peikert and Waters [PW08] suggest the composition of an LTDF generator, an ABO TDF generator, and a strongly unforgeable one-time signature scheme to achieve IND-CCA. Boldyreva, Fehr, and O'Neill [BFO08] give a similar construction (with the signature scheme replaced by a target-collision resistant hash function) that achieves PRIV-CCA for block-sources.

As pointed out above, this result does not lift generically to MMR1-CCA. Of course, it is possible that one or both of these constructions satisfy our stronger notion, but this requires a fresh proof.[12]

[11] Note that a family of trapdoor permutations is syntactically the same as a deterministic encryption scheme.

[12] In another direction, [RSV13] consider novel notions of LTDFs for their adaptive CCA setting.

It remains open to instantiate MMR-CCA in the standard model, but prior work suggests that LTDFs and ABO LTDFs are a promising approach.

$\mathsf{PtD}[F\text{-}\mathsf{DOAEP}]$ IS MM+IND-CCA. Security against MM attacks is achievable with a scheme that is both MMR1- and ANON-CCA via Theorem 3. Here we show that, under certain restrictions, instantiating the base scheme with F-DOAEP is MM-CCA assuming only that F is OWF secure.

Theorem 7 **($\mathsf{PtD}[F\text{-}\mathsf{DOAEP}]$ is MM+IND-CCA).** *Let* PKEAD *be defined by* $\mathsf{PtD}[F\text{-}\mathsf{DOAEP}]$ *with parameters* $n, k_0, k_1, \rho : \mathbb{N} \to \mathbb{N}$ *in Figure 7, where functions* H_1, H_2, *and* G *are modeled as random oracles. Suppose that* $n(k) \geq k_0(k) + k_1(k)$ *for all* k. *There exists an adversary* $\mathcal{B}$ *such that the following conditions hold:*

– *(MM-CCA) Let* $\mu_1, \mu_2, v : \mathbb{N} \to \mathbb{N}$ *be functions where* $\mu_2(k) \in \omega(\log k)$. *Let* $\mathcal{A}$ *be a* (μ_1, v)*-mm-adversary and* $\mathcal{R}$ *be a* (μ_2, v, ρ)*-r-source. Suppose that* $\mathcal{A}$ *makes exactly* q_e *queries to its* **LR** *oracle,* q_d *queries to its* **Dec** *oracle, and* q_1, q_2, *and* q_G *to oracles* H_1, H_2, *and* G *respectively. Then*

$$\mathbf{Adv}^{\text{mm-cca}}_{\mathsf{PKEAD},\mathcal{R}}(\mathcal{A}, k) \leq 2q_e v(k) \cdot \mathbf{Adv}^{\text{owf}}_{F}(\mathcal{B}, k) + \frac{5q_e v(k)(q_1 + q_d)}{2^{\mu_1(k)+\mu_2(k)-1}} + \frac{3q_e v(k)(q_G + q_d) + v(k)(q_2 + q_d) + 2q_d}{2^{k_0(k)-1}} + \frac{q_e(q_1 + q_d)^2}{2^{\rho(k)-1}} .$$

– *(IND-CCA) Let* $\mathcal{A}$ *be an adversary, which makes* q_e *queries to its* **LR** *oracle,* q_d *queries to its* **Dec** *oracle, and* q_1, q_2, *and* q_G *to oracles* H_1, H_2, *and* G *respectively. Then*

$$\mathbf{Adv}^{\text{mm-cca}}_{\mathsf{PKEAD},\mathcal{R}}(\mathcal{A}, k) \leq 2q_e \cdot \mathbf{Adv}^{\text{owf}}_{F}(\mathcal{B}, k) + \frac{6q_e q_d + 3q_e q_G + q_e q_2}{2^{k_0(k)-1}} + \frac{6q_e(q_1 + q_d)^2}{2^{\rho(k)-1}} .$$

In each case, we have $\mathsf{time}_{\mathcal{B}}(k) = \mathsf{time}_{\mathcal{A}}(k) + O(q_d q_1 q_G q_2)$.

Let us explain this claim a bit. (The proof can be found in Appendix C.5.) First, we only consider the case where $n \geq k_0 + k_1$. The designers of F-DOAEP give two bounds for its PRIV security [BBO07, Theorem 5.2]: one for inputs of length less than $k_0 + k_1$ and another for inputs of length greater than $k_0 + k_1$. The distinction arises from the fact that, in the former case, $\mathcal{A}$'s random oracle queries consist of strings less than k_1 bits in length. The problem is that $\mathcal{B}$ is looking for the preimage under f of its input y, which is a k_1-bit string. The solution is a lemma that relates the OWF advantage of $\mathcal{B}$ to the advantage of another inverter adversary whose task is to return a substring of the preimage rather than the whole string [BBO07, Lemma A.1]. (This is closely related to the POWF notion of [FOPS04].) We focus on the $n \geq k_0 + k_1$ case for simplicity.

Second, restricting the associated data space to strings of length k_0 ensures that the entropy contained in the message and the random padding is encoded by the right side of the input. This restriction is not strictly necessary to achieve security, but it allows us to appeal directly to the OWF security of the trapdoor permutation in the analysis. It is worth noting that the associated data is encrypted along with the message and randomizer, and that this is undesirable if the associated data is a long string. In practice, the associated data might actually be a hash of the associated data, but we emphasize that security is achieved only for the hash and *not* the associated data itself.

Remark. In section 6.1, we showed that F-$\mathsf{EME\text{-}OAEP}$, a variant of F-OAEP, is secure against MM attacks, but that its concrete security depends only on the entropy in the coins. Here we see that adding an additional Feistel round yields improved concrete security against MM attacks, since we are able to prove a bound for F-DOAEP that does take the message entropy into account. This would be the case even without restricting the messages and associated data as we have.

Randomized-then-Deterministic. The composition of a randomized and a deterministic encryption scheme suggested by Bellare et al. is defined in Figure 7. The idea is to first encrypt the message and associated data using a randomized scheme, then encrypt the result using a deterministic scheme. Security appeals to the randomized scheme when the coins are uniform and appeals to the deterministic scheme when the message-coins are only high min-entropy. The $\mathsf{RtD}[\Pi_r, \Pi_d]$ composition has message space determined by both Π_r and Π_d; the associated data is the same as for Π_r.

$\mathsf{RtD}[\Pi_r, \Pi_d]$ is MMR+IND-CCA. Let $\mathsf{PKEAD} = \mathsf{RtD}[\Pi_r, \Pi_d]$. It is clearly IND-CPA if Π_r is IND-CPA. Bellare et al. show that PKEAD, under certain conditions, is MMR-CPA if Π_d is MMR-CPA [BBN+09, theorem 6.2]. Their argument easily extends to the CCA setting, as shown below. In order to prove this composition works, it is necessary that the output of the randomized scheme Π_r has as much entropy as its inputs. The following property, formalized by [BBN+09], suffices for entropy-preserving encryption.

Injective encryption. A PKEAD scheme PKEAD with associated data space AD and randomness length $\rho(\cdot)$ is said to be *injective* if for every $k \in \mathbb{N}$, $(pk, sk) \in [\mathsf{PKEAD.Kgen}(1^k)]$, $H \in \mathsf{AD}$, and $(M, r), (M', r') \in \{0,1\}^* \times \{0,1\}^{\rho(k)}$, if $(M, r) \neq (M', r')$, then $\mathsf{PKEAD.Enc}^H_{pk}(M\,; r) \neq \mathsf{PKEAD.Enc}^H_{pk}(M'\,; r')$. This gives us two useful properties: one, if the equality pattern of $\boldsymbol{M}$ and $\boldsymbol{r}$ is distinct, then so is the equality pattern of $\mathsf{Enc}^H_{pk}(\boldsymbol{M}\,; \boldsymbol{r})$; two, if $\langle M, r\rangle$ has min-entropy $\mu(\cdot)$, then $C = \mathsf{Enc}^H_{pk}(M\,; r)$ has min-entropy $\mu(\cdot)$. Many schemes possess this property, including ElGamal [Elg85] and OAEP [BR95].

Theorem 8 **($\mathsf{RtD}[\Pi_r, \Pi_d]$ is MMR+IND-CCA).** *Let Π_r be an injective and randomized PKEAD scheme with associated data space AD and randomness length $\rho(\cdot)$, let Π_d be a deterministic PKE scheme, and let $\mathsf{PKEAD} = \mathsf{RtD}[\Pi_r, \Pi_d]$ as defined in Figure 7.*

– *(MMR-CCA) Let $\mu, v : \mathbb{N} \to \mathbb{N}$ be functions where $\mu(k) \in \omega(\log k)$. Let $\mathcal{A}$ be a (μ, v, ρ)-mmr adversary. There exists a $(\mu, v, 0)$-mmr adversary $\mathcal{B}$ such that for every k, it holds that $\mathbf{Adv}^{\text{mmr-cca}}_{\mathsf{PKEAD}}(\mathcal{A}, k) = \mathbf{Adv}^{\text{mmr-cca}}_{\Pi_d}(\mathcal{B}, k)$, where $\mathcal{B}$ has the same runtime as $\mathcal{A}$.*

– *(IND-CCA) Let $\mathcal{A}$ be an adversary. There exists an adversary $\mathcal{B}$ such that for every k, it holds that $\mathbf{Adv}^{\text{ind-cca}}_{\mathsf{PKEAD}}(\mathcal{A}, k) = \mathbf{Adv}^{\text{ind-cca}}_{\Pi_r}(\mathcal{B}, k)$, where $\mathcal{B}$ has the same runtime as $\mathcal{A}$.*

The proof is by a simple extension of [BBN+09, Theorem 6.2]; see Appendix C.6 for details. This result gives us a simple way to securely realize MMR+IND-CCA encryption, but we need to show how to instantiate the deterministic scheme Π_d. The same result we have for PtD applies here; if Π_d is a u-LTDF, then $\mathsf{RtD}[\Pi_r, \Pi_d,]$ is MMR-CPA for block-sources. Again, securely instantiating MMR-CCA in the standard model remains open.

$\mathsf{RtD}[\Pi_r, F\text{-}\mathsf{DOAEP}]$ is MM+IND-CPA. As before, we consider security against MM attacks when the deterministic scheme is F-DOAEP. MMR-CCA security is out of reach for this particular composition, as evidenced by an attack against the PRIV-CCA-security of $\mathsf{RSA\text{-}DOAEP}$ pointed out by [BBO07]. (Their attack can be carried out in the MM-CCA game.) Nonetheless, we show the following:

Theorem 9 **($\mathsf{RtD}[\Pi_r, F\text{-}\mathsf{DOAEP}]$ is MM+IND-CPA).** *Let F be a trapdoor permutation generator with randomness length $k_1(\cdot)$. Let F-DOAEP be the deterministic scheme defined in Figure 7 with parameters $k_0, k_1, n : \mathbb{N} \to \mathbb{N}$. Let Π be an injective PKEAD scheme with associated data space AD and randomness length $\rho(\cdot)$. Let $\mathsf{PKEAD} = \mathsf{RtD}[\Pi, F\text{-}\mathsf{DOAEP}]$ as defined in Figure 7, where H_1, H_2, and G are random oracles.*

– *(MM-CPA) Let $\mu_1, \mu_2, v : \mathbb{N} \to \mathbb{N}$ be functions where $\mu_2(k) \in \omega(\log k)$. Let $\mathcal{A}$ be a (μ_1, v)-mm-adversary and $\mathcal{R}$ be a (μ_2, v, ρ)-r-source. Suppose that $\mathcal{A}$ makes q_e queries to its **LR** oracle and q_1, q_2, and q_G to oracles H_1, H_2, and G respectively. Suppose that $n(k) < k_0(k)+k_1(k)$ for all k. Then there exists an adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\text{mm-cpa}}_{\mathsf{PKEAD},\mathcal{R}}(\mathcal{A}, k) \le q_e v(k) q_G \cdot \sqrt{\delta_2(k) + \mathbf{Adv}^{\text{owf}}_{F}(\mathcal{B}, k)} + q_e \delta_1(k) + \frac{4q_e v(k) \cdot q_1 q_G}{2^{\mu_1(k)+\mu_2(k)}} + \frac{4q_e v(k)(q_G + q_2)}{2^{k_0(k)}} \,,$$

where $\delta_c(k) = 2^{ck_1(k)-2c(n(k)-k_0(k))+5}$ and $\mathsf{time}_{\mathcal{B}}(k) = \mathsf{time}_{\mathcal{A}}(k) + O(\log v(k) + q_2 \log q_2 + k_1(k)^3)$. Suppose that $n(k) \ge k_0(k) + k_1(k)$ for all k. Then there exists an adversary $\mathcal{B}$ such that

$$\mathbf{Adv}^{\text{mm-cpa}}_{\mathsf{PKEAD},\mathcal{R}}(\mathcal{A}, k) \le q_e v(k) \cdot \mathbf{Adv}^{\text{owf}}_{F}(\mathcal{B}, k) + \frac{4q_e v(k) \cdot q_1 q_G}{2^{\mu_1(k)+\mu_2(k)}} + \frac{4q_e v(k)(q_G + q_2)}{2^{k_0(k)}}$$

and $\mathsf{time}_{\mathcal{B}}(k) = \mathsf{time}_{\mathcal{A}}(k) + O(\log v(k) + q_2 \log q_2)$.

– *(IND-CPA) Let $\mathcal{A}$ be an IND-CPA adversary. There exists an IND-CPA adversary $\mathcal{B}$ such that $\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{PKEAD}}(\mathcal{A}, k) = \mathbf{Adv}^{\text{ind-cpa}}_{\Pi}(\mathcal{B}, k)$ and $\mathcal{B}$ has the same run time as $\mathcal{A}$.*

The first part of the claim follows from an argument built upon the proof that $\mathsf{RSA\text{-}DOAEP}$ is PRIV secure [BBO07, Theorem 5.2]. Our results differ from theirs in the following way. In the PRIV experiment, the adversary is given the public key only *after* it submits its **LR** query. This means that the public key has entropy from the perspective of the adversary at this point in the game. This fact is used to bound the advantage $\mathcal{A}$ gets from its random oracle queries *before* it queries **LR**. This is why the inputs to the RO in the DOAEP construction are prepended with the public key. (See figure 7.) Because the adversary is given the public key in our setting, we must find another way to bound this advantage. Once we have done this, however, we can use their argument directly to obtain the claim. We refer the reader to Appendix C.7 for the full proof.

Acknowledgments

Christopher Patton and Thomas Shrimpton are supported by NSF grant CNS-1564446. Alexandra Boldyreva is supported in part by NSF grants CNS-1318511 and CNS-1422794. We thank Cihan Eryonucu for providing an initial survey of APIs during his visit to Georgia Tech. We also thank Mihir Bellare and the anonymous reviewers for valuable feedback, as well as Joseph Choi and Animesh Chhotaray for bug fixes.

References

ABF$^+$17. Yasemin Acar, Michael Backes, Sascha Fahl, Simon Garfinkel, Doowom Kim, Michelle L. Mazurek, and Christian Stransky. Comparing the usability of cryptographic apis. In *Proceedings of the 38th IEEE Symposium on Security and Privacy*, 2017.

ABP16. Michel Abdalla, Fabrice Benhamouda, and David Pointcheval. Public-key encryption indistinguishable under plaintext-checkable attacks. *IET Information Security*, 10(6), 2016.

BBDP01. Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In *ASIACRYPT 2001: Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security*, 2001.

BBN$^+$09. Mihir Bellare, Zvika Brakerski, Moni Naor, Thomas Ristenpart, Gil Segev, Hovav Shacham, and Scott Yilek. Hedged public-key encryption: How to protect against bad randomness. In *ASIACRYPT 2009: Proceedings of the 15th International Conference on the Theory and Application of Cryptology and Information Security*, 2009.

BBO07. Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. Deterministic and efficiently searchable encryption. In *CRYPTO 2007: Proceedings of the 27th Annual International Cryptology Conference on Advances in Cryptology*, 2007.

BCC$^+$13. Daniel J. Bernstein, Yun-An Chang, Chen-Mou Cheng, Li-Ping Chou, Nadia Heninger, Tanja Lange, and Nicko van Someren. Factoring RSA keys from certified smart cards: Coppersmith in the wild. In *ASIACRYPT 2013: Proceedings of the 19th International Conference on the Theory and Application of Cryptology and Information Security*, 2013.

BDPR98. Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In *CRYPTO 1998: Proceedings of the 18th Annual International Cryptology Conference*, 1998.

BFM15. Christina Brzuska, Pooya Farshim, and Arno Mittelbach. Random-oracle uninstantiability from indistinguishability obfuscation. In *TCC (2)*, pages 428–455. Springer, 2015.

BFO08. Alexandra Boldyreva, Serge Fehr, and Adam O'Neill. On notions of security for deterministic encryption, and efficient constructions without random oracles. In *CRYPTO 2008: Proceedings of the 28th Annual International Cryptology Conference*, 2008.

BH15. Mihir Bellare and Viet Tung Hoang. Resisting randomness subversion: Fast deterministic and hedged public-key encryption in the standard model. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, 2015.

BR93. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *CCS 1993: Proceedings of the 1st ACM Conference on Computer and Communications Security*, 1993.

BR95. Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In *EUROCRYPT 1994: Workshop on the Theory and Application of Cryptographic Techniques*, 1995.

BR06. Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In *EUROCRYPT 2006: Proceedings of the 24th Annual International Conference on The Theory and Applications of Cryptographic Techniques*, 2006.

Bro05. Daniel R. L. Brown. A weak-randomizer attack on RSA-OAEP with $e = 3$. Cryptology ePrint Archive, Report 2005/189, 2005. http://eprint.iacr.org/2005/189.

BT16. Mihir Bellare and Björn Tackmann. Nonce-based cryptography: Retaining security when randomness fails. In *EUROCRYPT 2016: Proceedings of the 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2016.

CCS09. Jan Camenisch, Nishanth Chandran, and Victor Shoup. A public key encryption scheme secure against key dependent chosen plaintext and adaptive chosen ciphertext attacks. In *EUROCRYPT 2009: Proceedings of the 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2009.

CMG$^+$16. Stephen Checkoway, Jacob Maskiewicz, Christina Garman, Joshua Fried, Shaanan Cohney, Matthew Green, Nadia Heninger, Ralf-Philipp Weinmann, Eric Rescorla, and Hovav Shacham. A systematic analysis of the Juniper Dual EC incident. In *CCS 2016: Proceedings of the 23rd ACM Conference on Computer and Communications Security*, 2016.

DGP09. Leo Dorrendorf, Zvi Gutterman, and Benny Pinkas. Cryptanalysis of the random number generator of the Windows operating system. *ACM Transactions on Information Systems and Security*, 13(1), 2009.

DK05. Yevgeniy Dodis and Jonathan Katz. Chosen-ciphertext security of multiple encryption. In *TCC 2005: Proceedings of the Second Theory of Cryptography Conference*, 2005.

DKN. Thai Duong, Emilia Kasper, and Quan Nguyen. Scaling Crypto Testing with Project Wycheproof. https://www.cs.bris.ac.uk/Research/CryptographySecurity/RWC/2017/thai.duong.pdf.

DPR$^+$13. Yevgeniy Dodis, David Pointcheval, Sylvain Ruhault, Damien Vergniaud, and Daniel Wichs. Security analysis of pseudo-random number generators with input: /dev/random is not robust. In *CCS 2013: Proceedings of the 20th ACM Conference on Computer and Communications Security*, 2013.

Elg85. Taher Elgamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO 1984: Proceedings of CRYPTO'84*, 1985.

FO99. Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *CRYPTO 1999: Proceedings of the 19th Annual International Cryptology Conference*, 1999.

FOPS04. Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, and Jacques Stern. RSA-OAEP is secure under the RSA assumption. *Journal of Cryptology*, 17(2), 2004.

GM05. Zvi Gutterman and Dahlia Malkhi. Hold your sessions: An attack on java session-id generation. In Alfred Menezes, editor, *CT-RSA 2005: The Cryptographers' Track at the RSA Conference 2005*, 2005.

HDWH12. Nadia Heninger, Zakir Durumeric, Eric Wustrow, and J. Alex Halderman. Mining your Ps and Qs: Detection of widespread weak keys in network devices. In *USENIX 2012: Proceedings of the 21st USENIX Conference on Security Symposium*, 2012.

HKOZ16. Viet Tung Hoang, Jonathan Katz, Adam O'Neill, and Mohammad Zaheri. Selective-opening security in the presence of randomness failures. In *ASIACRYPT 2016: Proceedings of the 22nd International Conference on the Theory and Application of Cryptology and Information Security*, 2016.

Kos01. Takeshi Koshiba. A new aspect for security notions: Secure randomness in public-key encryption schemes. In *PKC 2001: Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography*, 2001.

KOS10. Eike Kiltz, Adam O'Neill, and Adam Smith. Instantiability of RSA-OAEP under chosen-plaintext attack. In *CRYPTO 2010: Proceedings of the 30th Annual Conference on Advances in Cryptology*, 2010.

MKJR16. Kathleen Moriarty, Burt Kaliski, Jakob Jonsson, and Andreas Rusch. PKCS #1: RSA Cryptography Specifications Version 2.2. RFC 8017, 2016.

MMS13. Kai Michaelis, Christopher Meyer, and Jörg Schwenk. Randomly failed! The state of randomness in current Java implementations. In *CT-RSA 2013: The Cryptographers' Track at the RSA Conference 2013*, 2013.

Mue08. Markus Mueller. Debian OpenSSL predictable PRNG, 2008. https://www.exploit-db.com/exploits/5622/.

PSS14. Kenneth G. Paterson, Jacob C. N. Schuldt, and Dale L. Sibborn. Related randomness attacks for public key encryption. In *PKC 2014: Proceedings of the 17th International Conference on Practice and Theory in Public-Key Cryptography*, 2014.

PW08. Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In *STOC 2008: Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, 2008.

RSV13. Ananth Raghunathan, Gil Segev, and Salil Vadhan. Deterministic public-key encryption for adaptively chosen plaintext distributions. In *EUROCRYPT 2013: Proceedings of the 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2013.

RY10. Thomas Ristenpart and Scott Yilek. When good randomness goes bad: Virtual machine reset vulnerabilities and hedging deployed cryptography. In *NDSS 2010: Proceedings of the 17th Annual Network and Distributed System Security Symposium*, 2010.

Sho02. Victor Shoup. OAEP reconsidered. *Journal of Cryptology*, 15(4), 2002.

Sho04. Victor Shoup. ISO18033-2: An emerging standard for public-key encryption (final committee draft), 2004. http://shoup.net/iso.

Yil10. Scott Yilek. Resettable public-key encryption: How to encrypt on a virtual machine. In *CT-RSA 2010: Proceedings of the 10th Cryptographers' Track at the RSA Conference 2010*, 2010.

A Coins-guessing attacks

Let $\mathsf{PKEAD} = (\mathsf{Kgen}, \mathsf{Enc}, \mathsf{Dec})$ be a PKEAD scheme with associated data space AD and randomness length $\rho(\cdot)$. We assume PKEAD is reasonable in the following sense: for any pk, H, M, r, if $\mathsf{Enc}_{pk}^{H}(M\,; r) \neq \bot$, then for every M' such that $|M'| = |M|$, it holds that $\mathsf{Enc}_{pk}^{H}(M'\,; r) \neq \bot$. Associate to PKEAD a *message length* $n(\cdot)$ such that for every $k \in \mathbb{N}$, $(pk, sk) \in [\mathsf{Kgen}(1^k)]$, $H \in \mathsf{AD}$, and $M \in \{0,1\}^{n(k)}$, it holds that $\Pr[C \leftarrow\!\!\$\, \mathsf{Enc}_{pk}^{H}(M) : C \neq \bot] = 1$. Let $\mu_1, \mu_2 : \mathbb{N} \to \mathbb{N}$ be functions. Let $\mathcal{R}$ be a $(\mu_2, 1, \rho)$-r-source. Consider the following $(\mu_1, 1)$-mm-adversary $\mathcal{A}$ (defined formally in Figure 8) playing the MM-CPA game. Fix $k \in \mathbb{N}$ and let $n = n(k)$ and $\rho = \rho(k)$. Let $f : \{0,1\}^* \times \{\bot\} \to \{0,1\} \times \{\bot\}$ be a function and $r \in \{0,1\}^{\rho}$. For each $b \in \{0,1\}$, the messages

```
$\underline{\mathcal{A}_{f,r}^{\mathbf{LR},\mathbf{Dec}}(1^k, pk)}$
$\boldsymbol{H} \leftarrow\!\!\$\ \mathsf{AD}^v$
$\boldsymbol{C} \leftarrow\!\!\$\ \mathbf{LR}(\boldsymbol{H}, \mathcal{M})$
return $f(\boldsymbol{C}[1])$

$\underline{\mathcal{M}(1^k)}$
for each $b \in \{0,1\}$ do
  $\boldsymbol{M}_b[1] \leftarrow\!\!\$\ \{0,1\}^n$
  until $f(\mathsf{Enc}(pk, \boldsymbol{H}[1], \boldsymbol{M}_b[1];\, r)) = b$
return $(\boldsymbol{M}_0, \boldsymbol{M}_1)$
```

Fig. 8. Coins-guessing attack on the MM-CPA and -CCA games.

comprising $\boldsymbol{M}_b$ in the output of the mm-source specified by $\mathcal{A}$ are uniformly sampled from a set

$$X_b^H = \{M \in \{0,1\}^n : f(\mathsf{Enc}_{pk}^{H}(M;\, r)) = b\}$$

for some $H \in \mathsf{AD}$. The sets X_0^H and X_1^H are disjoint for every such H, and hence the source is distinct. Moreover, as long as $|X_b^H| \geq 2^{\mu_1(k)}$ for sufficiently large k, then the source has min-entropy $\mu_1(\cdot)$. This establishes that $\mathcal{A}$ is a valid adversary.

The adversary's strategy is to guess the coins selected for encryption by the $\mathbf{LR}$ oracle. It correctly outputs the challenge bit if it manages to guess the coins. Let r^* be a string in the range of $\mathcal{R}(1^k)$ such that $\Pr[r \leftarrow\!\!\$\, \mathcal{R}(1^k) : r = r^*] = 2^{-\mu_2(k)}$. (This is a sequence of coins with the most probability mass associated to it.) By guessing these coins and selecting a predicate f which partitions the message space roughly in half, the adversary gains advantage at least $2^{-\mu_2(k)}$. If $\mu_2(k) \in \omega(\log k)$ (i.e., has high min-entropy), then $\mathbf{Adv}_{\mathsf{PKEAD},\mathcal{R}}^{\text{mm-cpa}}(\mathcal{A}_{f,r^*}, k)$ is negligible in k. Suppose

$\mu_2(k) \notin \omega(\log k)$. Then there exists a constant c such that for sufficiently large k, it holds that $\mu_2(k) < c \cdot \log k$. Hence, the adversary's advantage is at least k^{-c}, which is non-negligible. Moreover, this attack is effective even if the messages have high min-entropy.

In light of this attack, we require the coin source to have high min-entropy. We remark that one can extend this attack to the MMR setting by having the adversary guess the public key instead of the coins. (The adversary specifies the coins in this setting; there is no need to guess them.) But since the key-generation algorithm is executed with uniform coins, the adversary's advantage is necessarily negligible.

Efficacy of coins guessing. The information leaked about the plaintexts by this attack is non-trivial. In order to carry out the attack, the adversary specifies a predicate f of the ciphertexts that roughly divides the message space in half: for example, the parity of the string or the least significant bit. (The latter is a good choice for RSA-based instantiations, since this bit is hardcore for RSA.) As pointed out by [BBN+09], the ciphertext is non-trivial information about the message. While the quantity $f(C)$ is not immediately interesting on its own, the predicate could be crafted so that it leaks information about the message corresponding to C itself.

One might argue that mounting this attack is not realistic, since it requires knowing the coins most likely to be output by the source, and this information may notbe forthcoming. Yet the adversary knows how the coins are generated (in particular, it knows the algorithm $\mathcal{R}$) and can sample from the source in order to gather statistics about its distribution. In fact, one could modify the attack in Figure 8 so that adversary executes its source with coins r sampled from $\mathcal{R}(1^k)$. One might then consider formalizing the idea that the coins source is *unknown* to the adversary, and that it is unable to sample from it. However, it is not clear that this can be done in our uniform model of computation. Instead, one might consider the frameworks of [Kos01, RSV13].

B Table of common crypto libraries

We include a complete list of the libraries we surveyed in Table 1. This is not meant to be an exhaustive list of all libraries available; we also did not attempt to assess which of these are the most popular.

C Proofs

This appendix contains full proofs for theorems stated in the body, but not proven. Corresponding figures can be found in Appendix D.

C.1 Theorem 1.

We lift the proof of [BBN+09, Theorem 5.2] to the CCA setting. Fix $k \in \mathbb{N}$. Without loss of generality, suppose that $\mathcal{A}$ makes its q queries to **LR** before querying **PKout**. Refer to the games defined in the top panel of Figure 9 associated to integer $i \in [0..q]$ and adversary $\mathcal{A}$. Let $\mathbf{H}_i(\mathcal{A})$ denote the output of the game given by the code *excluding* the boxed statements and $\mathbf{H}'_i(\mathcal{A})$ denote the output of the game *including* the boxed statements. In both games, the adversary's **LR** query is answered by first executing the message-coins source to get $(\boldsymbol{M}_0, \boldsymbol{M}_1, \boldsymbol{r})$. The first $q-i$ queries are answered by encrypting vector $\boldsymbol{M}_1$ and the remaining i queries are answered by encrypting vector $\boldsymbol{M}_0$. The difference between the games is that, in the latter, the i-th query is encrypted

Name	URL
Botan	https://botan.randombit.net
BoringSSL	https://boringssl.googlesource.com/boringssl
Bouncy Castle	https://bouncycastle.org
cryptlib	http://cryptlib.com
crypto++	https://www.cryptopp.com
go/crypto	https://golang.org/pkg/crypto
Libgcrypt	https://www.gnu.org/software/libgcrypt
LibreSSL	https://www.libressl.org
NaCl	https://nacl.cr.yp.to
OpenSSL	https://www.openssl.org
PyCrypto	https://www.dlitz.net/software/pycrypto
SCAPI	https://scapi.readthedocs.io/en/latest
wolfSSL	https://wolfssl.com

Table 1. A list of the crypto libraries surveyed for this paper.

using a different key. Let $h_i = \Pr[\,\mathbf{H}_i(\mathcal{A}) \Rightarrow 1\,]$ and $h'_i = \Pr[\,\mathbf{H}'_i(\mathcal{A}) \Rightarrow 1\,]$ for each $i \in [0..q]$. First, we observe that $h_0 = \Pr[\,\mathbf{Exp}^{\text{mmr-cca}}_{\mathsf{PKEAD}}(\mathcal{A}, k) \mid b = 1\,]$ and $h_q = \Pr[\,\neg\mathbf{Exp}^{\text{mmr-cca}}_{\mathsf{PKEAD}}(\mathcal{A}, k) \mid b = 0\,]$ where $b = 1$ (resp. $b = 0$) denotes the event that the challenge bit in $\mathcal{A}$'s game is 1 (resp. 0). It follows that

$$\begin{aligned}
\mathbf{Adv}^{\text{mmr-cca}}_{\mathsf{PKEAD}}(\mathcal{A}, k) &= 2 \cdot \Pr[\,\mathbf{Exp}^{\text{mmr-cca}}_{\mathsf{PKEAD}}(\mathcal{A}, k)\,] - 1 \\
&= \Pr[\,\mathbf{Exp}^{\text{mmr-cca}}_{\mathsf{PKEAD}}(\mathcal{A}, k) \mid b = 1\,] + \Pr[\,\mathbf{Exp}^{\text{mmr-cca}}_{\mathsf{PKEAD}}(\mathcal{A}, k) \mid b = 0\,] - 1 \\
&= \Pr[\,\mathbf{Exp}^{\text{mmr-cca}}_{\mathsf{PKEAD}}(\mathcal{A}, k) \mid b = 1\,] - \Pr[\,\neg\mathbf{Exp}^{\text{mmr-cca}}_{\mathsf{PKEAD}}(\mathcal{A}, k) \mid b = 0\,] \\
&= h_0 - h_q = \sum_{i=0}^{q-1}(h_i - h'_i) + \sum_{i=0}^{q-1}(h'_i - h'_{i+1}) + \sum_{i=0}^{q-1}(h'_{i+1} - h_{i+1}) \qquad (1) \\
&= \sum_{i=0}^{q-1}(h_i - h'_i) + \sum_{i=1}^{q}(h'_i - h_i) + \sum_{i=0}^{q-1}(h'_i - h'_{i+1}).
\end{aligned}$$

Refer to adversaries $\mathcal{D}_i$ and $\overline{\mathcal{D}}_i$ defined in the bottom-left panel of Figure 9. The former is defined by the code excluding the boxed statement and the latter is defined by the code including the boxed statement. The difference is that the former outputs the compliment of $\mathcal{A}$'s output. By construction, we have that

$$h_i = \Pr[\,\mathbf{Exp}^{\text{anon-cca}}_{\mathsf{PKEAD}}(\mathcal{D}_i, k) \mid d = 1\,] \text{ and } h'_i = \Pr[\,\neg\mathbf{Exp}^{\text{anon-cca}}_{\mathsf{PKEAD}}(\mathcal{D}_i, k) \mid d = 0\,]$$

for each $i \in [0..q-1]$. Similarly, for every $i \in [1..q]$ it holds that

$$h'_i = \Pr\big[\,\mathbf{Exp}^{\text{anon-cca}}_{\mathsf{PKEAD}}(\overline{\mathcal{D}}_i, k) \mid d = 1\,\big] \text{ and } h_i = \Pr\big[\,\neg\mathbf{Exp}^{\text{anon-cca}}_{\mathsf{PKEAD}}(\overline{\mathcal{D}}_i, k) \mid d = 0\,\big].$$

Let $\mathcal{D}$ be the adversary that first samples $d \leftarrow\!\!\$\ \{0,1\}$, then $j \leftarrow\!\!\$\ [0+d..(q-1)+d]$. If $d = 1$, then it executes $\overline{\mathcal{D}}_j$ with its own oracles; otherwise it executes $\mathcal{D}_j$. Finally, when $\mathcal{D}_j$ (resp. $\overline{\mathcal{D}}_j$) outputs g, output $d \oplus g$. Then

$$\begin{aligned}
&\sum_{i=0}^{q-1}(h_i - h'_i) + \sum_{i=1}^{q}(h'_i - h_i) = \\
&\quad \sum_{i=0}^{q-1} \big(\Pr[\,\mathbf{Exp}^{\text{anon-cca}}_{\mathsf{PKEAD}}(\mathcal{D}_i, k) \mid d = 1\,] - \Pr[\,\neg\mathbf{Exp}^{\text{anon-cca}}_{\mathsf{PKEAD}}(\mathcal{D}_i, k) \mid d = 0\,]\big) + \\
&\quad \sum_{i=1}^{q} \big(\Pr\big[\,\mathbf{Exp}^{\text{anon-cca}}_{\mathsf{PKEAD}}(\overline{\mathcal{D}}_i, k) \mid d = 1\,\big] - \Pr\big[\,\neg\mathbf{Exp}^{\text{anon-cca}}_{\mathsf{PKEAD}}(\overline{\mathcal{D}}_i, k) \mid d = 0\,\big]\big) \\
&\quad = 2q \cdot \mathbf{Adv}^{\text{anon-cca}}_{\mathsf{PKEAD}}(\mathcal{D}, k).
\end{aligned} \tag{2}$$

Finally, refer to $\mathcal{B}_i$ in the bottom-right panel of Figure 9. By construction, we have $h'_i = \Pr[\,\mathbf{Exp}^{\text{mmr-cca}}_{\mathsf{PKEAD}}(\mathcal{B}_i, k) \mid b = 1\,]$ and $h'_{i+1} = \Pr[\,\neg\mathbf{Exp}^{\text{mmr-cca}}_{\mathsf{PKEAD}}(\mathcal{B}_i, k) \mid b = 0\,]$ for each $i \in [0..q-1]$. Let $\mathcal{B}$ be the adversary that first samples $j \twoheadleftarrow_{\$} [0..q-1]$, then executes $\mathcal{B}_j$. Then

$$\begin{aligned}
\sum_{i=0}^{q-1}(h'_i - h'_{i+1}) &= \sum_{i=0}^{q-1} \big(\Pr[\,\mathbf{Exp}^{\text{mmr-cca}}_{\mathsf{PKEAD}}(\mathcal{B}_i, k) \mid b = 1\,] \\
&\quad - \Pr[\,\neg\mathbf{Exp}^{\text{mmr-cca}}_{\mathsf{PKEAD}}(\mathcal{B}_i, k) \mid b = 0\,]\big) \\
&= q \cdot \mathbf{Adv}^{\text{mmr-cca}}_{\mathsf{PKEAD}}(\mathcal{B}, k).
\end{aligned} \tag{3}$$

Putting together equations (1), (2), and (3) yields the claim.

C.2 Theorem 3.

Refer to the games $\mathbf{H}_b$ and $\mathbf{H}'_b$ defined in Figure 10 and parameterized by $b \in \{0, 1\}$. The latter is defined by the code including the boxed statements. The difference between them is that in the latter, the **LR** query is answered using pk_1 instead of pk_0. First, there exists an adversary $\mathcal{A}'$ such that

$$\mathbf{Adv}^{\text{mm-cca}}_{\mathsf{PKEAD},\mathcal{R}}(\mathcal{A}', k) = \Pr[\,\mathbf{H}_1(\mathcal{A}') \Rightarrow 1\,] - \Pr[\,\mathbf{H}_0(\mathcal{A}') \Rightarrow 1\,]. \tag{4}$$

It simply executes $\mathcal{A}$ on its own inputs and with access to its own oracles. Next, let $\mathcal{A}'$ be an adversary and let $h_b = \Pr[\,\mathbf{H}_b(\mathcal{A}') \Rightarrow 1\,]$ and $h'_b = \Pr[\,\mathbf{H}'_b(\mathcal{A}') \Rightarrow 1\,]$. Then $(h_1 - h_0) = (h_1 - h'_1) + (h'_1 - h'_0) + (h'_0 - h_0)$. Refer to adversary $\mathcal{D}_b$ defined in the bottom-left panel of Figure 10 and parameterized by $b \in \{0, 1\}$. Suppose without loss of generality that $\mathcal{A}'$ queries $\mathbf{PKout}'$ only after it queries $\mathbf{LR}'$. Let $p(x', y', d')$ denote the probability that adversary $\mathcal{D}_0$ outputs d' given that $x = x'$, $y = y'$, and $d = d'$ where d denotes the challenge bit in $\mathcal{D}_0$'s game. Then

$$\begin{aligned}
\mathbf{Adv}^{\text{anon-cca}}_{\mathsf{PKEAD}}(\mathcal{D}_0, k) &= 2 \cdot \Pr[\,\mathbf{Exp}^{\text{anon-cca}}_{\mathsf{PKEAD}}(\mathcal{D}_0, k)\,] - 1 \\
&= \Pr[\,\mathbf{Exp}^{\text{anon-cca}}_{\mathsf{PKEAD}}(\mathcal{D}_0, k) \mid d = 1\,] \\
&\quad + \Pr[\,\mathbf{Exp}^{\text{anon-cca}}_{\mathsf{PKEAD}}(\mathcal{D}_0, k) \mid d = 0\,] - 1 \\
4 \cdot \mathbf{Adv}^{\text{anon-cca}}_{\mathsf{PKEAD}}(\mathcal{D}_0, k) + 4 &= p(1,1,1) + p(1,0,1) + p(0,1,1) + p(0,0,1) \\
&\quad + p(1,1,0) + p(1,0,0) + p(0,1,0) + p(0,0,0).
\end{aligned} \tag{5}$$

By construction, it holds that $h'_0 = p(1,1,1) = p(1,0,0)$ and $(1 - h_0) = p(0,0,1) = p(0,1,0)$. Moreover, we have that $p(0,1,1) = 1 - p(0,0,0)$ and $p(1,1,0) = 1 - p(1,0,1)$. Putting this together with the previous equation yields

$$\begin{aligned}4 \cdot \mathbf{Adv}^{\text{anon-cca}}_{\mathsf{PKEAD}}(\mathcal{D}_0, k) + 4 &= 2h'_0 - 2(1 - h_0) + 2 \\ 2 \cdot \mathbf{Adv}^{\text{anon-cca}}_{\mathsf{PKEAD}}(\mathcal{D}_0, k) &= h'_0 - h_0.\end{aligned} \tag{6}$$

By a similar argument, we have that $2 \cdot \mathbf{Adv}^{\text{anon-cca}}_{\mathsf{PKEAD}}(\mathcal{D}_1, k) = h_1 - h'_1$. Next define $\mathcal{D}$ as the adversary which flips a coin $b \leftarrow\!\!\$\ \{0, 1\}$, executes $\mathcal{D}_b$, and returns whatever it outputs. It follows that

$$(h_1 - h'_1) + (h'_0 - h_0) = 4 \cdot \mathbf{Adv}^{\text{anon-cca}}_{\mathsf{PKEAD}}(\mathcal{D}, k). \tag{7}$$

To conclude the proof, we bound the quantity $(h'_1 - h'_0)$ with adversary $\mathcal{B}$ defined in the bottom-right panel of figure 10. Again, we assume that $\mathcal{A}'$ first queries $\mathbf{LR}'$ before it queries $\mathbf{PKout}'$. By construction, we have that

$$h'_1 - h'_0 = \mathbf{Adv}^{\text{mmr-cca}}_{\mathsf{PKEAD}}(\mathcal{B}, k). \tag{8}$$

Putting together equations (4), (7) and (8) yields the claim.

C.3 Theorem 5

First, by Theorem 2, there exists a (μ_1, v)-mm-adversary $\mathcal{A}_1$ that makes one $\mathbf{LR}$ query such that $\mathbf{Adv}^{\text{mm-cca}}_{\mathsf{PKEAD},\mathcal{R}}(\mathcal{A}, k) \le q_e \cdot \mathbf{Adv}^{\text{mm-cca}}_{\mathsf{PKEAD},\mathcal{R}}(\mathcal{A}_1, k)$. We bound $\mathcal{A}_1$'s advantage in the remainder.

We argue by game-rewriting, beginning with $\mathbf{Exp}^{\text{mm-cca}}_{\mathsf{PKEAD},\mathcal{R}}(\mathcal{A}_1, k)$ and ending with a game simulated by POWF adversary $\mathcal{B}$, which we specify. After each revision, we bound the advantage of $\mathcal{A}_1$ in distinguishing between neighboring games.

Fix $k \in \mathbb{N}$ and let $n = n(k)$, $\rho = \rho(k)$, $\tau = \tau(k)$, $\mu_2 = \mu_2(k)$, $v = v(k)$, and $m = m(k)$. Note that the encryption algorithm outputs $\perp$ only if the $\mathsf{pad}(M) = \perp$, where M is the input message. This occurs if the length of M is greater than $m - \tau - 8$ or is not a multiple of 8 (encryption is defined over byte strings.) In the remainder, we assume the source specified by the adversary's $\mathbf{LR}$ outputs string in the induced message space. This is without loss of generality, since otherwise the query would reject.

In the following, let $\mathbf{Adv}_i(\mathcal{A}_1, k) = 2 \cdot \Pr[\,\mathbf{G}_i(\mathcal{A}_1, k)\,] - 1$ for integer i. Refer to the top panel of Figure 11. Game $\mathbf{G}_0$ is defined by the code *including* the statements in red, but *excluding* the statements in green. This is just the MM-CCA game instantiated with PKEAD, and thus $\mathbf{Adv}^{\text{mmr-cca}}_{\mathsf{PKEAD},\mathcal{R}}(\mathcal{A}_1, k) = \mathbf{Adv}_0(\mathcal{A}_1, k)$.

Refer to game $\mathbf{G}_1$ in the same panel. It is defined by the code *including* the statements in green, but *excluding* the statements in red. In this step, the calls to G by the $\mathbf{LR}$ oracle are removed. Instead, the outputs of G corresponding to each $\boldsymbol{r}_i$ are generated in advance of the $\mathbf{LR}$ query. If the adversary manages to guess $\boldsymbol{r}_i$ for some $i \in [v]$ and asks $\mathsf{G}(\boldsymbol{r}_i)$, then it is given the correct output in response. Since this amounts to only a syntactic (and not a semantic) change, we have that $\mathbf{Adv}_0(\mathcal{A}_1, k) = \mathbf{Adv}_1(\mathcal{A}_1, k)$.

Now refer to the bottom-left panel of Figure 11. Game $\mathbf{G}_2$ is defined by the code excluding the statements in red. We claim that the outputs of $\mathbf{G}_1(\mathcal{A}_1, k)$ and $\mathbf{G}_2(\mathcal{A}_1, k)$ are identically distributed up to the setting of flag bad_1. Suppose that $\mathbf{G}_1(\mathcal{A}_1, k)$ never sets bad_1. Then from the perspective of the adversary, the value A_i, and hence $X_1 = X_0 \oplus A_i$, is a uniform-random string for each $i \in [v]$. The same is true if $\mathbf{G}_2(\mathcal{A}_1, k)$ never sets bad_1, except in this case we simply let $X_1 = A_i$. Since source $\mathcal{R}$ has min-entropy $\mu_2(\cdot)$, it holds that $\Pr[\,\mathbf{G}_1(\mathcal{A}_1, k)\,\mathsf{sets}\,\mathsf{bad}_1\,] \le v(q_G + q_d)/2^{\mu_2}$. Applying the fundamental lemma of game-playing [BR06], we conclude that

$$\Pr[\,\mathbf{G}_1(\mathcal{A}_1, k)\,] - \Pr[\,\mathbf{G}_2(\mathcal{A}_1, k)\,] \le v(q_G + q_d)/2^{\mu_2}. \tag{9}$$

As usual, game $\mathbf{G}_3$ (top panel of Figure 12) is defined by the code including the green statements and excluding the red ones. Similar to the rewriting of $\mathbf{G}_1$, we change the code so that the outputs of H_2 corresponding to each A_i are generated in advance of the **LR** query. We modify the H_2 oracle so that the adversary's queries are consistent. We have that $\mathbf{Adv}_2(\mathcal{A}_1, k) = \mathbf{Adv}_3(\mathcal{A}_1, k)$.

Game $\mathbf{G}_4$ is defined by the revisions in the bottom-right panel of Figure 11. This game excludes the statements in red. By an argument similar to the one closing the gap between games $\mathbf{G}_1$ and $\mathbf{G}_2$, it follows that games $\mathbf{G}_3$ and $\mathbf{G}_4$ are identical until the flag bad^* is set. Thus,

$$\Pr[\,\mathbf{G}_3(\mathcal{A}_1, k)\,] - \Pr[\,\mathbf{G}_4(\mathcal{A}_1, k)\,] \le \Pr[\,\mathbf{G}_3(\mathcal{A}_1, k)\ \mathsf{sets}\ \mathsf{bad}^*\,]\ . \tag{10}$$

Note that $\Pr[\,\mathbf{G}_4(\mathcal{A}_1, k)\,] = 1/2$, since its oracle queries are independent of the challenge bit. The reduction will relate the advantage of our OWF adversary $\mathcal{B}$ to the probability that bad^* gets set. But first, we need to attend to the decryption oracle.

We rewrite the decryption oracle (similar to the reduction of Fujisaki et al. [FOPS04]) so that it can be computed without having the trapdoor. This makes it possible for $\mathcal{B}$ to simulate $\mathcal{A}_1$'s decryption queries, since it is given f, but not f^{-1}. Game $\mathbf{G}_5$ (bottom panel of Figure 12) is defined by the code including the statements in green, but excluding the statements in red. The only semantic change is the condition under which the decryption oracle rejects. On input (H, C), instead of checking if $\mathsf{H}_1(H) = T$, it checks if T was ever output by a prior query to H_1. More precisely, if there exists a string H^* such that $T_1[H^*] = T$, then it accepts; otherwise it rejects. This subsumes the prior condition, since in the line above, the oracle is queried on H. Since each $\mathsf{H}_1(H^*)$ is a uniform-random, τ-bit string for each such H^*, the probability that this condition holds for a particular query (H, C) is at most $(q_1 + q_d)/2^\tau$. Summing over all queries to **Dec**, we have that

$$\Pr[\,\mathbf{G}_4(\mathcal{A}_1, k)\,] - \Pr[\,\mathbf{G}_5(\mathcal{A}_1, k)\,] \le q_d(q_1 + q_d)/2^\tau\ . \tag{11}$$

Refer now to the top-left panel of Figure 13. Game $\mathbf{G}_6$, defined by the code including the statement in green, differs only in the action taken after setting bad_5. The flag gets set if for some query $\mathbf{Dec}(H, C)$, the integrity check succeeds, but the corresponding input to oracle G was undefined. If this occurs, then the revised oracle outputs $\perp$. The probability that these conditions occur is at most the probability that the integrity check succeeds *given* that the input to G was undefined. In this case, the string X_0 is a uniform-random string, and the probability that $\mathsf{H}_1(H) = T$, where T is the last τ bits of X_0, is at most $1/2^\tau$. Summing over all decryption queries, the probability that bad_6 gets set is at most $q_d/2^\tau$. Then

$$\Pr[\,\mathbf{G}_5(\mathcal{A}_1, k)\,] - \Pr[\,\mathbf{G}_6(\mathcal{A}_1, k)\,] \le q_d/2^\tau\ . \tag{12}$$

Next, game $\mathbf{G}_7$ (top-right panel of Figure 13) is defined by the code including the statement in green. Similar to the prior revision, the game is identical to $\mathbf{G}_6$ up to the setting of bad_6. This happens if $\mathcal{A}_1$ ever asks (H, C) of **Dec** and the integrity check succeeds, the corresponding input to G is defined (bad_5' not set), but the input to H_2 is undefined (bad_6' set). If these conditions hold, then the revised oracle returns $\perp$. Since the adversary did not guess the input to H_2, string Y_0 is uniformly distributed and its preimage is unknown to the adversary. Then probability that $T_G[Y_0]$ is defined is bounded by the probability that G was previously invoked on Y_0, which is at most $(q_d + q_G)/2^\rho$. Summing over all decryption queries, the probability that bad_5 gets set is at most $q_d(q_d + q_G)/2^\rho \le (q_G + q_d)^2/2^\rho$, and

$$\Pr[\,\mathbf{G}_6(\mathcal{A}_1, k)\,] - \Pr[\,\mathbf{G}_7(\mathcal{A}_1, k)\,] \le (q_G + q_d)^2/2^\rho\ . \tag{13}$$

This concludes our revisions of the decryption oracle.

We now exhibit a POWF adversary, which simulates $\mathcal{A}_1$ playing $\mathbf{G}_7$. Refer to adversary $\mathcal{B}$ defined in the bottom panel of Figure 13. Adversary $\mathcal{B}$ is given as input $(1^k, f, y)$, where presumably $y = f(x)$ for some $x \in \{0,1\}^n$. It executes $\mathcal{A}_1$ on input $(1^k, \langle f \rangle)$ with simulated oracles $\mathbf{LR}'$ and $\mathbf{Dec}'$, which we define in a moment. After $\mathcal{A}_1$ finishes interacting with its oracles and outputs a bit, adversary $\mathcal{B}$ chooses at random a string x' from the set of $\mathcal{A}_1$'s queries to its H_2 oracle. Finally, it outputs x'. Adversary $\mathcal{B}$ flips a coin b before executing $\mathcal{A}_1$.

On input $(\boldsymbol{H}, \mathcal{M})$, oracle $\mathbf{LR}'$ computes a sequenc $\boldsymbol{C}$ where each $\boldsymbol{C}_i = f(P \,\|\, [0])$ for a randomly chosen $(n-8)$-bit string P. It chooses a random $j \in [v]$ and replaces $\boldsymbol{C}_j$ with its own input y. Finally, it outputs $\boldsymbol{C}$. The simulation is perfect as long as the last 8 bits of $x = f^{-1}(y)$ are 0. Since x is a uniform-random string in the POWF experiment, this occurs with probability 2^{-8}. This will incur a penalty in the concrete security bound. (Note that this loss is observed in the standard [MKJR16].)

On input (H, C), oracle $\mathbf{Dec}'$ checks to see if the corresponding plaintext is *known* to $\mathcal{A}_1$ by checking to see if the ciphertext can be reconstructed from its random oracle queries. This similar to the plaintext extractor described by [FOPS04], but we must also check $\mathcal{A}_1$'s prior queries to H_1. Let Q_1 denote the set of queries to oracle H_1, Q_G to oracle G, and Q_2 to oracle H_2. For each pair $(A, B) \in Q_G \times Q_2$, let $U = \mathsf{G}(A) \oplus B$ and $V = \mathsf{H}_2(B) \oplus A$. If $f(B \,\|\, V \,\|\, [0]) = C$ and there exists a string $H^* \in Q_1$ such that $\mathsf{H}_1(H^*) = T$, where T is the last τ bits of U, then return the corresponding plaintext; otherwise return $\perp$. This perfectly simulates the $\mathbf{Dec}$ oracle in game $\mathbf{G}_7$ since we rewrote the oracle to output $\perp$ if the adversary never queried its oracles the values that would otherwise be output by the decryption oracle.

Let $x = f^{-1}(y)$, where y denotes $\mathcal{B}$'s input. Adversary $\mathcal{B}$ wins only if adversary $\mathcal{A}_1$ asks x' of its H_2 oracle where x' is the m-bit prefix of x. This occurs only if bad^* gets set. Since $\mathcal{B}$'s simulation is perfect with probability 2^{-8}, string x' was chosen from Q_2 uniform-randomly, and j was chosen uniform-randomly from $[v]$, we have that

$$\mathbf{Adv}^{\mathrm{powf}}_{F,m}(\mathcal{B}, k) \geq \frac{1}{256 v q_2} \cdot \Pr[\, \mathbf{G}_7(\mathcal{A}_1) \text{ sets } \mathsf{bad}^* \,] \,. \tag{14}$$

Let $p_i = \Pr[\mathbf{G}_i(\mathcal{A}_1, k)]$, $b_i = \Pr[\mathbf{G}_i(\mathcal{A}_1, k) \text{ sets } \mathsf{bad}_i]$, and $b_i^* = \Pr[\mathbf{G}_i(\mathcal{A}_1, k) \text{ sets } \mathsf{bad}^*]$. First, observe that the probabilities that $\mathbf{G}_7(\mathcal{A}_1, k)$ and $\mathbf{G}_6(\mathcal{A}_1, k)$ set bad^* are equal if bad_5 does not get set. Hence,

$$\begin{aligned} b_7^* &\geq b_6^*(1 - b_6) = b_6^* - b_6^* b_6 \\ &\geq b_6^* - b_6^* b_6 - (1 - b_6^*) b_6 \\ &= b_6^* - b_6 \geq b_6^* - (p_6 - p_7) \,. \end{aligned}$$

Similarly for b_6^* and b_5^*. Applying equations (11), (12), (13), and (14) yields

$$\begin{aligned} 256 v q_2 \;\cdot\; \mathbf{Adv}^{\mathrm{powf}}_{F,m}(\mathcal{B}, k) b_7^* &\geq b_6^* - (p_6 - p_7) \\ &\geq b_5^* - (p_5 - p_6) - (p_6 - p_7) \\ &\geq b_4^* - (p_4 - p_5) - (p_5 - p_6) - (p_6 - p_7) \\ &\geq \Pr[\, \mathbf{G}_3(\mathcal{A}_1, k) \text{ sets } \mathsf{bad}^* \,] - \frac{(q_d + 1)(q_d + q_1)}{2^\tau} - \frac{(q_G + q_d)^2}{2^\rho} \\ &\geq \Pr[\, \mathbf{G}_3(\mathcal{A}_1, k) \text{ sets } \mathsf{bad}^* \,] - \frac{(q_1 + q_d)^2}{2^\tau} - \frac{(q_G + q_d)^2}{2^\rho} \,. \end{aligned}$$

Applying equations (9) and (10) yields the bound:

$$\begin{aligned}
256vq_2 \cdot \mathbf{Adv}^{\text{powf}}_{F,m}(\mathcal{B},k) &\geq p_3 - p_4 - \frac{(q_1+q_d)^2}{2^\tau} - \frac{(q_G+q_d)^2}{2^\rho} \\
&\geq p_2 - \frac{1}{2} - \frac{(q_1+q_d)^2}{2^\tau} - \frac{(q_G+q_d)^2}{2^\rho} \\
&\geq p_1 - \frac{v(q_G+q_d)}{2^{\mu_2}} - \frac{1}{2} - \frac{(q_1+q_d)^2}{2^\tau} - \frac{(q_G+q_d)^2}{2^\rho} \\
&\geq \Pr[\,\mathbf{G}_0(\mathcal{A}_1,k)\,] - \frac{v(q_G+q_d)}{2^{\mu_2}} - \frac{1}{2} - \frac{(q_1+q_d)^2}{2^\tau} - \frac{(q_G+q_d)^2}{2^\rho} \\
&= \frac{1}{2}\mathbf{Adv}^{\text{mm-cca}}_{\mathsf{PKEAD},\mathcal{R}}(\mathcal{A}_1,k) - \frac{v(q_G+q_d)}{2^{\mu_2}} - \frac{(q_1+q_d)^2}{2^\tau} - \frac{(q_G+q_d)^2}{2^\rho}\ .
\end{aligned}$$

To complete the proof, we need only to comment on the runtime of $\mathcal{B}$. Simulating the **LR** and $\mathbf{RO}_i$ oracles needs only constant overhead whereas iterating over all prior H_1, G, and H_2 queries requires $O(q_1 q_G q_2)$ time for each **Dec** query. It follows that $\mathsf{time}_{\mathcal{B}}(k) = \mathsf{time}_{\mathcal{A}}(k) + O(g_d q_1 q_G q_2)$.

C.4 Theorem 6

MMR-CCA SECURITY. Without a loss of generality we assume that $\mathcal{A}$ does not repeat its hash and decryption queries and also that it does not repeat $\mathcal{M}$ in its LR oracle queries. Again, we consider a sequence of games, which we present in Figures D and D. We will justify the following:

$$\begin{aligned}
&\Pr\left[\mathbf{Exp}^{\text{mmr-cca}}_{\mathsf{HE}}(\mathcal{A},k) \Rightarrow \mathsf{true}\right] \\
&= \Pr[\mathbf{G}_0 \Rightarrow \mathsf{true}] && (15)\\
&\leq \Pr[\mathbf{G}_1 \Rightarrow \mathsf{true}] + \Pr[\mathbf{G}_1 \ \mathsf{sets\,bad}_1] && (16)\\
&= \Pr[\mathbf{G}_2 \Rightarrow \mathsf{true}] + \Pr[\mathbf{G}_2 \ \mathsf{sets\,bad}_1] && (17)\\
&= \Pr[\mathbf{G}_3 \Rightarrow \mathsf{true}] + \Pr[\mathbf{G}_3 \ \mathsf{sets\,bad}_1] && (18)\\
&\leq \Pr[\mathbf{G}_4 \Rightarrow \mathsf{true}] + \Pr[\mathbf{G}_4 \ \mathsf{sets\,bad}_1] + \Pr[\mathbf{G}_4 \ \mathsf{sets\,bad}_2] && (19)\\
&\leq \Pr[\mathbf{G}_4 \Rightarrow \mathsf{true}] + \Pr[\mathbf{G}_4 \ \mathsf{sets\,bad}_1] + \frac{q_d}{2^{k_P}} && (20)\\
&\leq \Pr[\mathbf{G}_5 \Rightarrow \mathsf{true}] + \Pr[\mathbf{G}_5 \ \mathsf{sets\,bad}_1] + \Pr[\mathbf{G}_5 \ \mathsf{sets\,bad}_3] + \frac{q_d}{2^{k_P}} && (21)\\
&= \Pr[\mathbf{G}_6 \Rightarrow \mathsf{true}] + \Pr[\mathbf{G}_6 \ \mathsf{sets\,bad}_1] + \Pr[\mathbf{G}_6 \ \mathsf{sets\,bad}_3] + \Pr[\mathbf{G}_6 \ \mathsf{sets\,bad}_4] + \frac{q_d}{2^{k_P}} && (22)\\
&\leq \Pr[\mathbf{G}_7 \Rightarrow \mathsf{true}] + \Pr[\mathbf{G}_7 \ \mathsf{sets\,bad}_1] + \Pr[\mathbf{G}_7 \ \mathsf{sets\,bad}_3] + \Pr[\mathbf{G}_7 \ \mathsf{sets\,bad}_4] \\
&+ \ \frac{q_d + q^2 v^2(k)}{2^{k_P}} && (23)\\
&\leq \Pr[\mathbf{G}_7 \Rightarrow \mathsf{true}] + \Pr[\mathbf{G}_7 \ \mathsf{sets\,bad}_1] + v(k)q(\mathbf{Adv}^{owf}_{F}(\mathcal{B},k) + \mathbf{Adv}^{auth}_{\mathsf{AEAD}}(\mathcal{D},k)) \\
&+ \ \frac{q_d + q^2 v^2(k)}{2^{k_P}} && (24)\\
&\leq \Pr[\mathbf{G}_8 \Rightarrow \mathsf{true}] + \Pr[\mathbf{G}_8 \ \mathsf{sets\,bad}_1] \\
&+ \ 2v(k)q(\mathbf{Adv}^{owf}_{F}(\mathcal{B},k) + \mathbf{Adv}^{ind-cpa}_{\mathsf{AEAD}}(\mathcal{C},k) + \mathbf{Adv}^{auth}_{\mathsf{AEAD}}(\mathcal{D},k)) + \frac{q_d + q^2 v^2(k)}{2^{k_P}} && (25)
\end{aligned}$$

$$\begin{aligned}
&\leq 1/2 + \Pr[\mathbf{G}_8 \ \ \mathsf{sets}\ \mathsf{bad}_1] + 2v(k)q(\mathbf{Adv}_F^{owf}(\mathcal{B},k) + \mathbf{Adv}_{\mathsf{AEAD}}^{ind-cpa}(\mathcal{C}) + \mathbf{Adv}_{\mathsf{AEAD}}^{auth}(\mathcal{D})) \\
&+ \frac{q_d + q^2v^2(k)}{2^{k_P}} \qquad (26) \\
&\leq 1/2 + \Pr[\mathbf{G}_9 \ \ \mathsf{sets}\ \mathsf{bad}_1] + \Pr[\mathbf{G}_9 \ \ \mathsf{sets}\ \mathsf{bad}_5] \\
&+ 2v(k)q(\mathbf{Adv}_F^{owf}(\mathcal{B},k) + \mathbf{Adv}_{\mathsf{AEAD}}^{ind-cpa}(\mathcal{C}) + \mathbf{Adv}_{\mathsf{AEAD}}^{auth}(\mathcal{D})) + \frac{q_d + q^2v^2(k)}{2^{k_P}} \qquad (27) \\
&\leq 1/2 + \Pr[\mathbf{G}_9 \ \ \mathsf{sets}\ \mathsf{bad}_1] + 2v(k)q(\mathbf{Adv}_F^{owf}(\mathcal{B},k) + \mathbf{Adv}_{\mathsf{AEAD}}^{ind-cpa}(\mathcal{C},k) + \mathbf{Adv}_{\mathsf{AEAD}}^{auth}(\mathcal{D})) \\
&+ \frac{q_d + q^2v^2(k)}{2^{k_P}} + \frac{q_d}{2^r} \qquad (28) \\
&= 1/2 + \Pr[\mathbf{G}_9 \ \ \mathsf{sets}\ \mathsf{bad}_1^b \vee \mathbf{G}_9 \ \ \mathsf{sets}\ \mathsf{bad}_1^a] \\
&+ 2v(k)q(\mathbf{Adv}_F^{owf}(\mathcal{B},k) + \mathbf{Adv}_{\mathsf{AEAD}}^{ind-cpa}(\mathcal{C}) + \mathbf{Adv}_{\mathsf{AEAD}}^{auth}(\mathcal{D})) + \frac{q_d + q^2v^2(k)}{2^{k_P}} + \frac{q_d}{2^r} \qquad (29) \\
&\leq 1/2 + \Pr[\mathbf{G}_{10} \ \ \mathsf{sets}\ \mathsf{bad}_1^b] + \Pr[\mathbf{G}_{10} \ \ \mathsf{sets}\ \mathsf{bad}_1^a] \\
&+ 2v(k)q(\mathbf{Adv}_F^{owf}(\mathcal{B},k) + \mathbf{Adv}_{\mathsf{AEAD}}^{ind-cpa}(\mathcal{C}) + \mathbf{Adv}_{\mathsf{AEAD}}^{auth}(\mathcal{D})) + \frac{q_d + q^2v^2(k)}{2^{k_P}} + \frac{q_d}{2^r} \qquad (30) \\
&\leq \frac{1}{2} + \frac{q_{\mathsf{H}_1}}{2^r} + \frac{(q_{\mathsf{H}_1} + q^2v(k))}{2^\mu} + \frac{q_d + q^2v^2(k)}{2^{k_P}} + \frac{q_d}{2^r} \\
&+ v(k)q(\mathbf{Adv}_F^{owf}(\mathcal{B},k) + \mathbf{Adv}_{\mathsf{AEAD}}^{ind-cpa}(\mathcal{C}) + \mathbf{Adv}_{\mathsf{AEAD}}^{auth}(\mathcal{D}))\ . \qquad (31)
\end{aligned}$$

Game $\mathbf{G}_0$ is like experiment $\mathbf{Exp}_{\mathsf{PKEAD}}^{\text{mmr-cca}}(\mathcal{A},k)$. Recall that hashes are modeled as random oracles, which means that there are extra procedures $\mathsf{H}_1, \mathsf{H}_2$ that need to be called whenever an algorithm of the scheme or the adversary have to compute a hash value. Also note that in $\mathbf{Exp}_{\mathsf{PKEAD}}^{\text{mmr-cca}}(\mathcal{A},k)$ the value X is picked at random according to the encryption algorithm. $\mathbf{G}_0$, however gets X by running $\mathcal{M}$. This models that the randomness source may be corrupted, and does not really change the distribution of the outputs. This justifies Equation (15).

Game $\mathbf{G}_1$ is like $\mathbf{G}_0$ except it sets the flag bad_1 when the answer to a query H_1 on some $\langle f^* \parallel R^*, H^*, X^*, M^* \rangle$ is already defined (i.e., the same input was queried before) and the queried public key matches the public key in the experiment, i.e. $R^* = R$, $f^* = f$. Notice that unless $\mathbf{G}_1$ sets bad_1, the output distribution of it is the same as that of $\mathbf{G}_0$. By the fundamental lemma [BR06], we have Equation (16).

Game $\mathbf{G}_2$ is like $\mathbf{G}_1$ except that K_P is picked uniformly at random. This does not change the distribution of the output of the game, because the absence of the "crucial" random oracle query described in $\mathbf{G}_1$ insures that K_P is independent from the view of $\mathcal{A}$. This justifies Equation (17).

Game $\mathbf{G}_3$ differs from $\mathbf{G}_2$ only in how the decryption queries are answered. If C_1^* in the attacker's decryption query is new, in that it is different from the asymmetric parts of all challenge ciphertexts, and random oracle query H_1 with result y has been made so that $f(y) = C_1^*$, then then the queried ciphertext is decrypted without knowing the secret key f^{-1}, as shown in $\mathbf{G}_3$. This change does not affect the output distribution of the game, since the ciphertexts are still decrypted as they should. This justifies Equation (18).

Game $\mathbf{G}_4$ is like $\mathbf{G}_3$ but there is an extra modification to the decryption oracle. If C_1^* in the attacker's decryption query is new, in that it is different from the asymmetric parts of all challenge ciphertexts, and no random oracle query H_1 with result y has been made so that $f(y) = C_1^*$, then the ciphertext is rejected and the event bad_2 is set if the queried ciphertext is valid. Notice that the changes do affect the distributions of the output of the games $\mathbf{G}_4$ and $\mathbf{G}_3$, unless a valid ciphertext has been rejected. But without the aforementioned hash query, the ciphertext can only be valid if $\mathcal{A}$ guessed the output of the hash, which can happen with probability at most $q_d/2^{k_P}$. This Equations (19) and (20).

Game $\mathbf{G}_5$ is like $\mathbf{G}_4$ but it sets the event bad_3 if $\mathcal{A}$ queries H_2 on $\langle f^* \parallel R^*, H^*, K_P^* \rangle$, where K_P^* has been used by the encryption algorithm as the input to f, and $R^* = R$, $f^* = f$. Unless bad_3 is set, the distributions of the outputs of the games are the same. This justifies Equation (21).

Game $\mathbf{G}_6$ is like $\mathbf{G}_5$ but it sets the event bad_4 and rejects if the asymmetric part of the ciphertext queried to the decryption oracle **Dec** is the same as the asymmetric part of one of the challenge ciphertexts returned by the **LR** oracle, and the whole ciphertext is valid. Note that according to the decryption algorithm of the scheme, the former condition guarantees that H_2 query is repeated. I.e., the decryption oracle calls H_2 on $\langle f^* \parallel R^*, H^*, K_P^* \rangle$, where K_P^* has been used by the encryption algorithm as the input to f, and $R^* = R$, $f^* = f$. We claim that unless bad_4 is set, the distributions of the outputs of the games $\mathbf{G}_6$ and $\mathbf{G}_5$ are the same. The only difference is that a valid ciphertext with "old" asymmetric part is rejected. This justifies Equation (22).

Game $\mathbf{G}_7$ is like $\mathbf{G}_6$ but K is picked uniformly at random for each query. This does not change the distribution of the output of the game, because the absence of the "crucial" random oracle and decryption oracle queries described in $\mathbf{G}_6$ insures that K is independent from the view of $\mathcal{A}$. In addition, the game sets bad if, while encrypting any message, the associated data and the asymmetric part C_1 of the ciphertext repeat. Note that this implies the repeating N as it is derived from the pair using the $\mathsf{extract}$ function. Since C_1 is computed by applying f to a randomly chosen K_p we can bound the probability of bad by using the birthday bound $q^2 v^2(k)/2^{k_P+1}$. This justifies Equation (23).

At this point we bound the probabilities that $\mathbf{G}_7$ sets bad_3 and bad_4. Notice that either bad_3 or bad_4 comes first (they cannot happen at the same time). Therefore, we first bound the probability of bad_3 in the absence of bad_4 and vice versa.

So assume event bad_3 happens (and bad_4 did not happen prior to that.) We have that $\mathcal{A}$ at some point queries H_2 on $\langle f^* \parallel R^*, H^*, K_P^* \rangle$, where K_P^* has been used as the input to f during the challenge ciphertexts calculation, and $R^* = R$, $f^* = f$. In this case, we can construct the one-wayness adversary $\mathcal{B}$ for F. $\mathcal{B}$ is given (f, y) as input. $\mathcal{B}$ would simulate $\mathbf{G}_7$ for $\mathcal{A}$ according to the game's description, except that $\mathcal{B}$ would provide its own challenge f as part of $\mathcal{A}$'s public key and the following change. At the beginning, $\mathcal{B}$ picks an index j at random from the number of LR queries $\mathcal{A}$ makes, and an index l at random from the number of messages in $\mathcal{A}$ queries. To simulate l's ciphertext C_1 in j's LR query (with some H'), $\mathcal{B}$ uses its own challenge y. Note that $\mathcal{B}$ cannot decrypt a decryption query $y \parallel C_2^*$, for any C_2^*, by computing $y^{-1}(C_1^*)$, since it does not have the corresponding secret key. In this case, $\mathcal{B}$ simply rejects. This is fine, since the absence of bad_3 means such a decryption query cannot be valid. Notice that $\mathcal{B}$ does not need the secret key to decrypt decryption queries $C_1^* \parallel C_2^*$, where C_1^* is new in that it does not coincide with the asymmetric parts of any challenge ciphertext and C_2^* is arbitrary. From $\mathbf{G}_3$ and $\mathbf{G}_4$ we know that the random oracle query H_1 with result h has been made so that $f(h) = C_1^*$ (verifying this only needs knowledge of f), so $\mathcal{B}$ can use h to decrypt the ciphertext. Moreover, $\mathcal{B}$ does not need the secret key to decrypt decryption queries $C_1^* \parallel C_2^*$, where C_1^* is part of some challenge ciphetext and $C_1^* \neq y$. This is because $\mathcal{B}$ picked K_p such that $f(K_p) = C_1^*$ by itself. Finally, whenever $\mathcal{A}$ makes the crucial H_2 query containing K_P', then $\mathcal{B}$ can output K_P'. Since j, l are independent from $\mathcal{A}$'s view, we have the upper bound on the probability of bad_3 as $v(k)q\mathbf{Adv}_F^{owf}(\mathcal{B}, k)$.

Now we consider the case when event bad_4 happens (and bad_3 did not happen prior to that.) This means that $\mathcal{A}$ queried the decryption oracle **Dec** on a ciphertext whose asymmetric part is the same as the asymmetric part of one of the challenge ciphertexts returned by the **LR** oracle, and the whole ciphertext is valid. The absence of bad_3 guarantees that the keys for the symmetric AEAD scheme are chosen at random during encryption. In this case, we can construct an adversary $\mathcal{D}$ breaking authenticity of the AEAD scheme. $\mathcal{D}$ can simulate the game according to the description, except it will use its own encryption oracle to simulate a (random) one of the challenge ciphertexts (the symmetric part) for $\mathcal{A}$. To answer each decryption query $(H^*, C_1^* \parallel C_2^*)$ made by $\mathcal{A}$, where $(H^*, C_1^* \parallel C_2^*)$ is new, but C_1^* is old, $\mathcal{D}$ will query its own special decryption oracle on $(\mathsf{extract}(\langle H^*, C_1^* \rangle), H^*, C_2^*)$. If it returns 0, then $\mathcal{D}$ rejects. If it returns 1, then $\mathcal{D}$ wins, since we know that either H^* or C_2^* is new. Note that the rest of the decryption queries can be answered according to the game description, as $\mathcal{D}$ knows all the symmetric decryption keys. Hence Equation (24) follows.

Game $\mathbf{G}_8$ is like $\mathbf{G}_7$ but all messages are picked uniformly at random. The difference between the probabilities of games $\mathbf{G}_8$ and $\mathbf{G}_7$ returning 1 can be upper-bounded by $v(k)q\mathbf{Adv}_{\mathsf{AEAD}}^{ind-cpa}(\mathcal{C}, k)$ for some $\mathcal{C}$ attacking ind-cpa security of the AEAD scheme. The reduction relies on the fact that the keys for AEAD are picked independently from $\mathcal{A}$'s view. We would actually construct a multi-user $\mathcal{C}'$ who would be given multiple left-right encryption oracles to simulate all LR queries by $\mathcal{A}$. The bound follows from the relation between the single-user and multi-user security of encryption. The decryption queries are answered as they should according to the game description. Note that the only problem could have been to answer the decryption queries corresponding to the secret symmetric keys underlying the ciphertexts created by the LR oracle of $\mathcal{C}'$. This could have came up if $\mathcal{A}$ made a decryption oracle query $C_1^* \parallel C_2^*$ where C_1 equals to one of the asymmetric parts of $\mathcal{A}$'s challenge ciphertexts simulated by $\mathcal{C}'$. But this situation is excluded by bad_4 set by $\mathbf{G}_6$. Also note that $\mathcal{C}'$ is nonce-respecting, since the absence of bad in $\mathbf{G}_7$ ensures that pairs (H, C_1) do not repeat, and this implies that IVs do not repeat.

Similarly, the difference between the probabilities of games $\mathbf{G}_8$ and $\mathbf{G}_7$ setting bad_1 can be upper-bounded by $v(k)q\mathbf{Adv}_{\mathsf{AEAD}}^{ind-cpa}(\mathcal{C}, k)$ for some $\mathcal{C}'$ attacking ind-cpa security of the AEAD scheme. $\mathcal{C}'$ is constructed similarly to $\mathcal{C}'$, except it returns 1 if bad_1 is set and 0 otherwise. The above justifies Equation (25). The factor of 2 is coming from from the standard argument, when considering two adversaries.

The probability that game $\mathbf{G}_8$ returns 1 ($\mathcal{A}$ has guessed the challenge bit correctly) is at most 1/2, because the view of $\mathcal{A}$ is independent from the challenge bit b. Thus we get Equation (26).

Game $\mathbf{G}_9$ is like $\mathbf{G}_8$ but there is an extra modification to the decryption oracle. $\mathbf{G}_9$ sets event bad_5 and rejects if $\mathcal{A}$ makes a decryption query $C_1^* \parallel C_2^*$, C_1^* is new in that it was not part of any challenge ciphertext, the public key has not been released, but the hash query H_2 on $\langle f^* \parallel R^*, H^*, K_P^* \rangle$, where $f(K_P^*) = C_1^*$, and $R^* = R$, $f^* = f$ has been made. Notice that the choice of R in the public key is independent of all hash queries made before the public key is revealed and bad_1^b is set. Also, in all cases when the public key is not released and different from those setting bad_5, the decryption oracle rejects, so the answers of the decryption oracle do not leak any information about the public key. Hence to set bad_5 the attacker must have guessed R in the public key, which can happen with probability at most $q_d/2^r$. This justifies Equations (27) and (28).

It only remains to bound the probability that $\mathbf{G}_9$ sets bad_1. The event is set when the answer to a query H_1 is already defined. The analysis is similar to that of the proof of Theorem 6.1 of [BBN+09]. There the decryption oracle queries are not considered, but in our case the decryption oracle does not make H_1 queries. We consider the cases when the crucial H_1 query is made before and after the public key is revealed. More precisely, we consider two sub-cases of event bad_1. We say that bad_1^b is set if a collision occurs before the public key is revealed, and bad_1^a is set after the public key is revealed. Game $\mathbf{G}_{10}$ is like $\mathbf{G}_9$, but it specifies these sub-cases. Hence we have Equation (29) and (30).

Let us first look at bad_1^b, which is set if a collision occurs before the public key is revealed. This can happen if $\mathcal{M}$ or $\mathcal{A}$ makes a query to H_1 oracle before the public key is released, and this query contains the public key $f \,\|\, R$. We observe that the choice of R is independent of all hash queries made before the public key is revealed and bad_1^b is set. Also, all query responses are independent of the outputs of $\mathcal{M}$. It is important that in $\mathbf{G}_9$ the decryption oracle answers do not leak any information about the public key before the public key is released (as all decryption oracle queries are rejected). Therefore, to set bad_1^b the attacker must have guessed R in the public key, which can happen with probability at most $q_{\mathsf{H}_1}/2^r$.

We note that, similarly to [BBN+09], we could also consider anonymity of F to bound the attacker's ability to guess the public key. But since our instantiation for F, the RSA, is not anonymous, we omit this option from our analysis.

If the attacker makes the crucial query after it learned the public key, we have to bound the probability of the event that a query of $\mathcal{A}$ to H_1 collides with a prior query done by $\mathcal{M}$ or two queries done by $\mathcal{M}$ collide. We are mainly interested in colliding M and X parts of the queries. Note that $\mathcal{A}$ does not learn the coins the source is run on and recall our assumptions that $\mathcal{A}$ does not repeat hash queries and sources in its LR queries. Following the proof of Theorem 6.1 of [BBN+09] we have

$$\Pr[\mathbf{G}_{10} \ \text{ sets } \mathsf{bad}_1^a] \leq \sum_{1 \leq u \leq q} \frac{q_{\mathsf{H}_1} + (q-1)v}{2^\mu} \leq \frac{(q_{\mathsf{H}_1} + q^2 v)}{2^\mu} \ .$$

This justifies Equation (31).

IND-CCA SECURITY. The proof of IND-CCA security is similar to (and simpler than) the above proof of MMR-CCA security, so we omit some details. Again, we consider a sequence of games associated with $\mathcal{A}$.

$\mathbf{G}_0$ is the "find-then-guess" IND-CCA experiment, where a random bit is flipped, the adversary is given the public key and the left-right encryption oracle and the decryption oracle. The former oracle takes inputs an associated data and two messages of equal length, and returns encryption of the message determined by the challenge bit. The latter oracle takes and associated data and a ciphertext and returns the decryption, unless the ciphertext was previously returned by the left-right encryption oracle for the associated data. The game returns 1 iff $\mathcal{A}$ correctly guesses the challenge bit.

$\mathbf{G}_1$ is different in that it sets event bad_1 if the answer to a random oracle query H_1 contains the public key and is already defined (i.e., the same input was queried before. Recall that we assume that the adversary does not repeat queries. Observe that to make such random oracle query the adversary should guess the (perfect) randomness X, and this can happen with probability at most $q_{\mathsf{H}_1}/2^{\rho(k)}$.

$\mathbf{G}_2$ sets K_p at random. In the absence of the crucial H_1 query this does not change the view of the adversary and the distribution of the game's output.

$\mathbf{G}_3$ sets bad_2 if $\mathcal{A}$ queries random oracle H_2 on an input containing the public key and K_p, where K_P^* has been used by the LR encryption oracle as the input to f.

$\mathbf{G}_4$ sets K at random. If $\mathbf{G}_4$ sets bad_2, we can construct an adversary $\mathcal{B}$ breaking one-wayness of F. The construction is basically the same as in the proof of MMR-CCA security of the scheme. $\mathcal{B}$ is given (f, y) and simulates $\mathbf{G}_4$ for $\mathcal{A}$. In particular, $\mathcal{B}$ picks an index at random and plugs in its challenge y as the asymmetric ciphertext part of the reply to corresponding left-right encryption query by $\mathcal{A}$. To answer a decryption oracle query $(H^*, C_1^* \,\|\, C_2^*)$ without the knowledge of the secret key, $\mathcal{B}$ does the following. If C_1^* is new in that it was not part of any challenge ciphertext, then $\mathcal{B}$ can decrypt the ciphertext without the secret key, if a query $\mathsf{H}_2(pk, H^*, K_p)$ has been made and $f(K_p) = C_1^*$. In this case $\mathcal{B}$ can use the result of this query as the key for AEAD. If no such query has been made, then $\mathcal{B}$ rejects. A valid ciphertext can be rejected this way only with probability at most $q_d/2^{k_P}$.

If $C_1^* \neq y$ and is old in that it was part of some challenge ciphertext, then $\mathcal{B}$ can properly decrypt. If $C_1^* = y$ then $\mathcal{B}$ rejects and in case a valid ciphertext is rejected, we can construct an adversary breaking authentication of the AEAD. Again, this is for simplicity, as $\mathcal{B}$ could properly decrypt the ciphertext even in this case.

In $\mathbf{G}_5$ random messages are encrypted. To bound the difference between $\mathbf{G}_5$ and $\mathbf{G}_4$ we can construct an IND-CPA adversary for AEAD. Similar to the proof for MMR-CCA, bound $qv(k)/2^{k_P-1}$ is needed to ensure that the adversary is nonce-respecting. Clearly, $\mathbf{G}_5$ outputs 1 with probability at most 1/2 as the challenge bit is independent from the view of $\mathcal{A}$.

Following the justifications similar to those of MMR-CCA proof, we can argue that

$$\begin{aligned}\mathbf{Adv}^{\text{ind-cca}}_{\mathsf{HE}}(\mathcal{A}, k) \leq 2v(k)q(\mathbf{Adv}^{owf}_{F}(\mathcal{B}, k) + \mathbf{Adv}^{ind-cpa}_{\mathsf{AEAD}}(\mathcal{C}, k) + \mathbf{Adv}^{auth}_{\mathsf{AEAD}}(\mathcal{D}, k))\\ + \frac{q_{\mathsf{H}_1}}{2^{\rho(k)-1}} + \frac{q_d + qv(k)}{2^{k_P-1}}\ .\end{aligned}$$

C.5 Theorem 7

We prove that this scheme is MM-CCA; its IND-CCA security follows from a nearly identical argument. By Theorem 2, there exists a (μ_1, v)-mm-adversary $\mathcal{A}_1$ that makes one **LR** query such that $\mathbf{Adv}^{\text{mm-cca}}_{\mathsf{PKEAD},\mathcal{R}}(\mathcal{A}, k) \leq q_e \cdot \mathbf{Adv}^{\text{mm-cca}}_{\mathsf{PKEAD},\mathcal{R}}(\mathcal{A}_1, k)$. We bound $\mathcal{A}_1$'s advantage in the remainder.

Fix $k \in \mathbb{N}$ and let $\mu_1 = \mu_1(k)$, $\mu_2 = \mu_2(k)$, $v = v(k)$, $n = n(k)$, $k_0 = k_0(k)$, $k_1 = k_1(k)$, and $\rho = \rho(k)$. Note that encryption rejects (outputs $\perp$) only if $\mathsf{pad}_{n-k_0}(\langle M, r\rangle) = \perp$ or $|H| \neq k_0$. In the following, we assume that $\mathcal{A}_1$'s **LR** query is a pair $(\boldsymbol{H}, \mathcal{M})$ where for $|\boldsymbol{H}[i]| = k_0$ for each i and for each $\boldsymbol{M}_b[i]$ in the output of $\mathcal{M}$, it holds that $\mathsf{pad}_{n-k_0}(\langle \boldsymbol{M}_b[i], r\rangle) \neq \perp$, where r is a ρ-bit string output by $\mathcal{R}$. This is without loss of generality, since otherwise the oracle would reject.

We argue by game rewriting, beginning with the MM-CCA game instantiated with PKEAD, $\mathcal{R}$, and $\mathcal{A}_1$, and ending with a game simulated by the OWF adversary $\mathcal{B}$, which we specify. In the following, let $\mathbf{Adv}_i(\mathcal{A}_1, k) = 2 \cdot \Pr[\,\mathbf{G}_i(\mathcal{A}_1, k)\,] - 1$. If bad_i is a flag, then let bad_i denote the event that $\mathbf{G}_i(\mathcal{A}_1, k)$ sets bad_i.

First, refer to game $\mathbf{G}_0$ in the bottom panel of Figure 16. It is defined by the code including the *red* statements and excluding the *green* statements. This is exactly the experiment $\mathbf{Exp}^{\text{mm-cca}}_{\mathsf{PKEAD},\mathcal{R}}(\mathcal{A}_1, k)$, and so $\mathbf{Adv}^{\text{mm-cca}}_{\mathsf{PKEAD},\mathcal{R}}(\mathcal{A}_1, k) = \mathbf{Adv}_0(\mathcal{A}_1, k)$. Game $\mathbf{G}_1$ (in the same panel) is defined by the code including the *green* statements and excluding the *red* statements. This game is revised so that the random oracle queries invoked by the **LR** oracle are replaced by random strings generated on the fly. The random oracles are instrumented so that its outputs are consistent with these values. This is only a syntactic change, and so $\mathbf{Adv}_0(\mathcal{A}_1, k) = \mathbf{Adv}_1(\mathcal{A}_1, k)$. In the next step, this revision will allow us to bound the advantage the adversary gets from its random oracle queries *before* making its **LR** query.

Refer to $\mathbf{G}_2$ defined in the bottom-left panel of Figure 17. It is identical to $\mathbf{G}_1$ until at least one of $\mathsf{bad}_{1,1}$, $\mathsf{bad}_{1,2}$, or $\mathsf{bad}_{1,3}$ is set. Let $\mathsf{bad}_{1,i}$ denote the event that $\mathbf{G}_1(\mathcal{A}_1, k)$ sets $\mathsf{bad}_{1,i}$. Then $\Pr[\,\mathbf{G}_1(\mathcal{A}_1, k)\,] - \Pr[\,\mathbf{G}_2(\mathcal{A}_1, k)\,] \leq \Pr[\,\mathsf{bad}_{1,1}\,] + \Pr[\,\mathsf{bad}_{1,2}\,] + \Pr[\,\mathsf{bad}_{1,3}\,]$. We upper bound each of these probabilities in turn. But first, let us consider the dependencies between each of these probabilities.

$$\begin{aligned}\Pr[\mathsf{bad}_{1,2}] &= \Pr[\,\mathsf{bad}_{1,2} \mid \mathsf{bad}_{1,1}\,]\Pr[\,\mathsf{bad}_{1,1}\,] + \Pr\left[\,\mathsf{bad}_{1,2} \mid \overline{\mathsf{bad}}_{1,1}\,\right]\Pr\left[\,\overline{\mathsf{bad}}_{1,1}\,\right] \\ &= \Pr[\,\mathsf{bad}_{1,2} \mid \mathsf{bad}_{1,1}\,]\Pr[\,\mathsf{bad}_{1,1}\,] \\ &+ \Pr\left[\,\mathsf{bad}_{1,2} \mid \overline{\mathsf{bad}}_{1,1}\,\right] - \Pr\left[\,\mathsf{bad}_{1,2} \mid \overline{\mathsf{bad}}_{1,1}\,\right]\Pr[\,\mathsf{bad}_{1,1}\,] \\ &\le \Pr[\,\mathsf{bad}_{1,2} \mid \mathsf{bad}_{1,1}\,]\Pr[\,\mathsf{bad}_{1,1}\,] + \Pr\left[\,\mathsf{bad}_{1,2} \mid \overline{\mathsf{bad}}_{1,1}\,\right] \\ &\le \Pr[\,\mathsf{bad}_{1,1}\,] + \Pr\left[\,\mathsf{bad}_{1,2} \mid \overline{\mathsf{bad}}_{1,1}\,\right] \;.\end{aligned}$$

Next, notice that $\Pr[\,\mathsf{bad}_{1,3}\,] = \sum_A \Pr[\,\mathsf{bad}_{1,3} \mid A\,]\Pr[\,A\,]$, where A is one of

$$\{\mathsf{bad}_{1,2} \cap \mathsf{bad}_{1,1}, \overline{\mathsf{bad}}_{1,2} \cap \mathsf{bad}_{1,1}, \mathsf{bad}_{1,2} \cap \overline{\mathsf{bad}}_{1,1}, \overline{\mathsf{bad}}_{1,2} \cap \overline{\mathsf{bad}}_{1,1}\} \;.$$

First, $\Pr[\,\mathsf{bad}_{1,2} \cap \mathsf{bad}_{1,1}\,] = \Pr[\,\mathsf{bad}_{1,2} \mid \mathsf{bad}_{1,1}\,]\Pr[\,\mathsf{bad}_{1,1}\,] \le \Pr[\,\mathsf{bad}_{1,1}\,]$. Second, $\Pr\left[\,\overline{\mathsf{bad}}_{1,2} \cap \mathsf{bad}_{1,1}\,\right] = \Pr\left[\,\overline{\mathsf{bad}}_{1,2} \mid \mathsf{bad}_{1,1}\,\right]\Pr[\,\mathsf{bad}_{1,1}\,] \le \Pr[\,\mathsf{bad}_{1,1}\,]$. Third,

$$\begin{aligned}\Pr\left[\,\mathsf{bad}_{1,2} \cap \overline{\mathsf{bad}}_{1,1}\,\right] &= \Pr\left[\,\mathsf{bad}_{1,2} \mid \overline{\mathsf{bad}}_{1,1}\,\right]\Pr\left[\,\overline{\mathsf{bad}}_{1,1}\,\right] \\ &= \Pr\left[\,\mathsf{bad}_{1,2} \mid \overline{\mathsf{bad}}_{1,1}\,\right](1 - \Pr[\,\mathsf{bad}_{1,1}\,]) \\ &\le \Pr\left[\,\mathsf{bad}_{1,2} \mid \overline{\mathsf{bad}}_{1,1}\,\right] \;.\end{aligned}$$

Thus, $\Pr[\,\mathsf{bad}_{1,3}\,] \le 2\Pr[\,\mathsf{bad}_{1,1}\,] + \Pr\left[\,\mathsf{bad}_{1,2} \mid \overline{\mathsf{bad}}_{1,1}\,\right] + \Pr\left[\,\mathsf{bad}_{1,3} \mid \overline{\mathsf{bad}}_{1,2} \cap \overline{\mathsf{bad}}_{1,1}\,\right]$. Summing this all together, we have

$$\begin{aligned}\Pr[\,\mathbf{G}_1(\mathcal{A}_1,k)\,] - \Pr[\,\mathbf{G}_2(\mathcal{A}_1,k)\,] \le 4\Pr[\,\mathsf{bad}_{1,1}\,] + 2\Pr\left[\,\mathsf{bad}_{1,2} \mid \overline{\mathsf{bad}}_{1,1}\,\right] + \\ + \Pr\left[\,\mathsf{bad}_{1,3} \mid \overline{\mathsf{bad}}_{1,2} \cap \overline{\mathsf{bad}}_{1,1}\,\right] \;.\end{aligned}$$

Consider the probability that $\mathsf{bad}_{1,1}$ get set. This occurs if for some $i \in [v]$, the adversary manages to guess the input $pk \,\|\, \boldsymbol{X}_r[i]$ to oracle H_1, where $\boldsymbol{X}_r[i]$ is a string corresponding to its future **LR** query. Since $\boldsymbol{X}_r[i]$ encodes both the message and coins, and $\mathcal{M}$ and $\mathcal{R}$ have min entropy $\mu_1(\cdot)$ and $\mu_2(\cdot)$ respectively, this occurs with probability at most $v/2^{\mu_1+\mu_2}$. Summing over all of $\mathcal{A}_1$'s H_1 and **Dec** queries (since **Dec** invokes H_1), we have that $\Pr[\,\mathsf{bad}_{1,1}\,] \le v(q_1+q_d)/2^{\mu_1+\mu_2}$.

Next, we bound the second term. If $\overline{\mathsf{bad}}_{1,1}$, then for every $i \in [v]$, string $\boldsymbol{r}_1[i]$ is a uniform-random, k_0-bit string. Thus, the probability that the adversary guesses the input $pk \,\|\, \boldsymbol{S}_0[i]$ to G before making its **LR** query is at most $v(q_G+q_d)/2^{k_0}$, since $\boldsymbol{S}_0[i] = \boldsymbol{r}_1[i] \oplus \boldsymbol{X}_\ell[i]$. Hence, $\Pr\left[\,\mathsf{bad}_{1,2} \mid \overline{\mathsf{bad}}_{1,1}\,\right] \le v(q_G+q_d)/2^{k_0}$.

Finally, a similar argument yields $\Pr\left[\,\mathsf{bad}_{1,3} \mid \overline{\mathsf{bad}}_{1,2} \cap \overline{\mathsf{bad}}_{1,1}\,\right] \le v(q_2+q_d)/2^{k_0}$. We conclude that

$$\Pr[\,\mathbf{G}_1(\mathcal{A}_1,k)\,] - \Pr[\,\mathbf{G}_2(\mathcal{A}_1,k)\,] \le \frac{4v(q_1+q_d)}{2^{\mu_1+\mu_2}} + \frac{2v(q_G+q_d)+v(q_2+q_d)}{2^{k_0}} \;. \tag{32}$$

In the next few steps, we revise the game so that $\mathcal{A}_1$'s random oracle queries are independent of its **LR** query. Game $\mathbf{G}_3$ (top-right panel of Figure 17) is identically distributed to $\mathbf{G}_2$ as long as bad_2 does not get set. This occurs if for some $i \in [v]$, the adversary manages to guess the input $pk \,\|\, \boldsymbol{X}_r[i]$ to H_1 where $\boldsymbol{X}_r[i]$ corresponds to its prior **LR** query. Since this string encodes the message and coins, we have that

$$\Pr[\,\mathbf{G}_2(\mathcal{A}_1,k)\,] - \Pr[\,\mathbf{G}_3(\mathcal{A}_1,k)\,] \le \frac{v(q_1+a_d)}{2^{\mu_1+\mu_2}} \;. \tag{33}$$

Next, game $\mathbf{G}_4$, defined in the bottom-left panel, is identically distributed to $\mathbf{G}_3$ as long as bad_4 does not get set. This occurs if the adversary asks $\mathsf{G}(X)$ where $X = pk \,\|\, \boldsymbol{S}_0[i]$ and $\boldsymbol{S}_0[i]$ coincides with its prior **LR** query for some $i \in [v]$. Since $\boldsymbol{S}_0[i]$ is a uniform-random, k_0-bit string for all such i, it holds that

$$\Pr[\,\mathbf{G}_3(\mathcal{A}_1, k)\,] - \Pr[\,\mathbf{G}_4(\mathcal{A}_1, k)\,] \le \frac{v(q_G + q_d)}{2^{k_0}}. \tag{34}$$

Finally, game $\mathbf{G}_5$, defined in the bottom-right panel, is identically distributed to $\mathbf{G}_3$ as long as bad^* does not get set. Then

$$\Pr[\,\mathbf{G}_4(\mathcal{A}_1, k)\,] - \Pr[\,\mathbf{G}_5(\mathcal{A}_1, k)\,] \le \Pr[\,\mathbf{G}_4(\mathcal{A}_1, k)\ \mathsf{sets\ bad}^*\,]\ . \tag{35}$$

We will relate the probability of this event to the advantage of adversary $\mathcal{B}$. Note that since the output of its queries are independent of the challenge bit, adversary $\mathcal{A}_1$ has no advantage in $\mathbf{G}_5$. Hence, $\Pr[\,\mathbf{G}_5(\mathcal{A}_1, k)\,] = 1/2$.

First, we revise the **Dec** oracle so that decryption can be simulated without possession of the trapdoor. (This is similar to the argument for Theorem 5.) Refer to the top-left panel of Figure 18.Game $\mathbf{G}_6$ is semantically the same as $\mathbf{G}_5$ so that $\mathbf{Adv}_5(\mathcal{A}_1, k) = \mathbf{Adv}_6(\mathcal{A}_1, K)$.

Game $\mathbf{G}_7$ is defined by the code in the top-right panel. This game is identical to $\mathbf{G}_6$ until bad_6 gets set. This occurs if for some query to $\mathbf{Dec}(H, C)$, the integrity check succeeded ($H = H^*$), but the corresponding input to the H_2 oracle was undefined before the query was made (bad_6' gets set). If these conditions hold, then the revised game outputs $\perp$. The probability that bad_6 gets set is at most the probability that the integrity check succeeds *given* that bad_6' gets set, which occurs if for some query $\mathbf{Dec}(H, C)$, it holds that $H = S_1 \oplus r_1 \oplus r_2$ where r_1 and r_2 denote the outputs of H_1 and H_2 respectively and S_1 is a string incident to C. Since r_2 is a uniform-random, k_0-bit string (who's pre-image under H_2 is unknown to $\mathcal{A}_1$), this is at most $1/2^{k_0}$. Summing over all decryption queries, we have that

$$\Pr[\,\mathbf{G}_6(\mathcal{A}_1, k)\,] - \Pr[\,\mathbf{G}_7(\mathcal{A}_1, k)\,] \le q_d/2^{k_0}\ . \tag{36}$$

Refer now to the bottom-right panel of Figure 17. Similarly, game $\mathbf{G}_8$ is identical to $\mathbf{G}_7$ until bad_7 gets set. This occurs if the for some decryption query (H, C), the integrity check passes, the input to H_2 was defined (bad_6' not set), but the input to H_1 was *not* defined (bad_7' set). If all these conditions hold, the revised game outputs $\perp$. This happens with probability at most the probability that the integrity check passes, given that the input to H_2 was defined, but the input to H_1 was not defined. This is equal to the probability that $H = S_1 \oplus r_1 \oplus r_2$. Since r_1 is a uniform-random, k_0-bit string (who's pre-image under H_1 is unknown to $\mathcal{A}_1$), this is at most $1/2^{k_0}$. Then

$$\Pr[\,\mathbf{G}_7(\mathcal{A}_1, k)\,] - \Pr[\,\mathbf{G}_8(\mathcal{A}_1, k)\,] \le q_d/2^{k_0}\ . \tag{37}$$

Next, refer to game $\mathbf{G}_9$ defined in the bottom-right panel of Figure 17. It is identical to $\mathbf{G}_8$ up to the setting of bad_8. This occurs if for some decryption query (H, C), the integrity check succeeds, the corresponding inputs to H_1 and H_2 were defined, but the input to G was undefined. As usual, if all of these conditions hold, then the revised game outputs $\perp$. The probability that all of these conditions hold is at most the probability that the input to H_1 was defined given that the integrity check passed, the input to H_2 was defined, and (crucially) the input to G was undefined. Let r_G denote the output of the G query. This is a uniform-random, $(n - k_0)$-bit string, who's pre-image under G is unknown to the adversary. The input to H_1 is $pk \,\|\, (r_G \oplus T_0)$, and so the probability that H_1 was defined at this point is bounded by the number of $\mathcal{A}_1$'s times H_1 was invoked, which is $q_1 + q_d$. Summing over $\mathcal{A}_1$'s decryption queries, and since $n - k_0 \geq \rho$ by construction, we have

$$\Pr[\,\mathbf{G}_8(\mathcal{A}_1, k)\,] - \Pr[\,\mathbf{G}_9(\mathcal{A}_1, k)\,] \leq q_d(q_1 + q_d)/2^\rho \ . \tag{38}$$

We now specify the OWF adversary, which simulates the execution of $\mathcal{A}_1$ in the $\mathbf{G}_9$ game. Refer to the top panel of Figure 16. Adversary $\mathcal{B}$ is given as input $(1^k, f, y)$ where y is presumably in the range of f. It chooses a random $j \leftarrow\!\!{\scriptstyle\$}\ [v]$, then executes $\mathcal{A}_1$ on input $(1^k, \langle f \rangle)$ and with access to oracles $\mathbf{LR}'$, $\mathbf{Dec}'$, H_1, G, and H_2, each of which it defines. On input $(\boldsymbol{H}, \mathcal{M})$, oracle $\mathbf{LR}'$ generates v random, n-bit strings. It then applies f to the last k_1 bits of each string. It replaces the last k_1 bits of the j-th string with its input y. On input (H, C), oracle $\mathbf{Dec}'$ checks to see if the plaintext is *known* to $\mathcal{A}_1$ by checking if it can be reconstructed from its random oracle queries. If the corresponding associated data is H, then it returns the plaintext. (This is similar to the plaintext extractor of [FOPS04].) Finally, the random oracles are defined in the usual way, except that H_2 has an extra check; if $\mathcal{A}_1$ ever asks $\mathsf{H}_2(X)$ such that the last k_1 bits of X are a string x' such that $f(x') = y$, it lets $x = x'$. When $\mathcal{A}_1$ finishes interacting with its oracles and outputs its guess, adversary $\mathcal{B}$ outputs x. (In case this value never gets set by H_2, it simply outputs $\bot$.)

Consider the event that $\mathbf{G}_9(\mathcal{A}_1, k)$ sets bad^*. Recall that this occurs if $\mathcal{A}_1$ ever asks $\mathsf{H}_2(X)$ where $X = pk \,\|\, \boldsymbol{T}_0[i]$ where $\boldsymbol{T}_0[i]$ is a string incident to its prior $\mathbf{LR}$ query. Since $\mathcal{B}$'s simulation of $\mathbf{G}_9$ is perfect, and since j is chosen uniformly, it follows that

$$\mathbf{Adv}^{\mathrm{owf}}_{F}(\mathcal{B}, k) \geq \frac{1}{v} \cdot \Pr[\,\mathbf{G}_9(\mathcal{A}_1, k)\ \mathsf{sets\ bad}^*\,] \ . \tag{39}$$

Applying equations (35), (36), (37), and (38),

$$\begin{aligned}
v \cdot \mathbf{Adv}^{\mathrm{owf}}_{F}(\mathcal{B}, k) &\geq \Pr[\,\mathbf{G}_4(\mathcal{A}_1, k)\ \mathsf{sets\ bad}^*\,] - \frac{2q_d}{2^{k_0}} - \frac{q_d(q_1 + q_d)}{2^\rho} \\
&\geq \Pr[\,\mathbf{G}_4(\mathcal{A}_1, k)\ \mathsf{sets\ bad}^*\,] - \frac{2q_d}{2^{k_0}} - \frac{(q_1 + q_d)^2}{2^\rho} \\
&\geq \Pr[\,\mathbf{G}_4(\mathcal{A}_1, k)\,] - \frac{1}{2} - \frac{2q_d}{2^{k_0}} - \frac{(q_1 + q_d)^2}{2^\rho} \ .
\end{aligned}$$

Finally, applying equations (32), (33), and (34) yields the bound:

$$\begin{aligned}
v \cdot \mathbf{Adv}^{\mathrm{owf}}_{F}(\mathcal{B}, k) &\geq \Pr[\,\mathbf{G}_3(\mathcal{A}_1, k)\,] - \frac{v(q_G + q_d)}{2^{k_0}} - \frac{1}{2} - \frac{2q_d}{2^{k_0}} - \frac{(q_1 + q_d)^2}{2^\rho} \\
&\geq \Pr[\,\mathbf{G}_2(\mathcal{A}_1, k)\,] - \frac{v(q_1 + q_d)}{2^{\mu_1 + \mu_2}} - \frac{v(q_G + q_d)}{2^{k_0}} - \frac{1}{2} - \frac{2q_d}{2^{k_0}} - \frac{(q_1 + q_d)^2}{2^\rho} \\
&\geq \Pr[\,\mathbf{G}_1(\mathcal{A}_1, k)\,] - \frac{4v(q_1 + q_d)}{2^{\mu_1 + \mu_2}} - \frac{2v(q_G + q_d) + v(q_2 + q_d)}{2^{k_0}} \\
&\quad - \frac{v(q_1 + q_d)}{2^{\mu_1 + \mu_2}} - \frac{v(q_G + q_d)}{2^{k_0}} - \frac{1}{2} - \frac{2q_d}{2^{k_0}} - \frac{(q_1 + q_d)^2}{2^\rho} \\
&\geq \Pr[\,\mathbf{G}_0(\mathcal{A}_1, k)\,] - \frac{1}{2} - \frac{5v(q_1 + q_d)}{2^{\mu_1 + \mu_2}} \\
&\quad - \frac{3v(q_G + q_d) + v(q_2 + q_d) + 2q_d}{2^{k_0}} - \frac{(q_1 + q_d)^2}{2^\rho} \\
&\geq \frac{1}{2} \cdot \mathbf{Adv}^{\text{mm-cca}}_{\mathsf{PKEAD},\mathcal{R}}(\mathcal{A}_1, k) - \frac{5v(q_1 + q_d)}{2^{\mu_1 + \mu_2}}
\end{aligned}$$

$$- \frac{3v(q_G + q_d) + v(q_2 + q_d) + 2q_d}{2^{k_0}} - \frac{(q_1 + q_d)^2}{2^\rho} .$$

To complete the proof, we need only attend to the runtime of $\mathcal{B}$. Each of the oracles can be simulated with constant overhead, except for the decryption oracle. Executing $\mathbf{Dec}'$ requires testing, in the worst case, every tuple in the set $Q_1 \times Q_G \times Q_2$. It follows that $\mathsf{time}_{\mathcal{B}}(k) = \mathsf{time}_{\mathcal{A}_1}(k) + O(q_d q_1 q_G q_2)$.

C.6 Theorem 8

Fix $k \in \mathbb{N}$. Let $\Pi_r = (\mathcal{K}_r, \mathcal{E}_r, \mathcal{D}_r)$ and $\Pi_d = (\mathcal{K}_d, \mathcal{E}_d, \mathcal{D}_d)$ with input length $n(\cdot)$. We start with the first claim. Adversary $\mathcal{B}$ simulates $\mathcal{A}$ in its game instantiated with PKEAD as follows. First, let $Q = \emptyset$, $\mathsf{pkout} = \mathsf{false}$, and execute $(pk_r, sk_r) \leftarrow\!\!\$\ \mathcal{K}(1^k)$. Next, execute $\mathcal{A}$ on input of 1^k. When $\mathcal{A}$ asks $\mathbf{LR}(\boldsymbol{H}, \mathcal{M})$, if $\mathsf{pkout} = \mathsf{true}$, then $\mathcal{B}$ outputs $\lightning$; otherwise it asks $\boldsymbol{C} \leftarrow\!\!\$\ \mathbf{LR}(\boldsymbol{H}, \mathcal{M}')$ where $\mathcal{M}'$ is a $(\mu, v, 0)$-mmr-source defined as follows: execute $(\boldsymbol{M}_0, \boldsymbol{M}_1, \boldsymbol{r}) \leftarrow\!\!\$\ \mathcal{M}(1^k)$. For each $b \in \{0, 1\}$ and $i \in [v(k)]$, let $\boldsymbol{X}_b[i] = \mathsf{pad}_n(\mathcal{E}_r(pk_r, \boldsymbol{H}[i], \boldsymbol{M}_b[i]; \boldsymbol{r}[i]))$ and $\boldsymbol{y}[i] = \varepsilon$. Return $(\boldsymbol{X}_0, \boldsymbol{X}_1, \boldsymbol{y})$. To finish the query, adversary $\mathcal{B}$ adds $(\boldsymbol{H}[i], \boldsymbol{C}[i])$ to Q for each $i \in [v(k)]$ and returns $\boldsymbol{C}$ to $\mathcal{A}$. When $\mathcal{A}$ asks $\mathbf{Dec}(H, C)$, if $(H, C) \in Q$, then return $\lightning$; otherwise, adversary $\mathcal{B}$ asks $X \leftarrow \mathbf{Dec}(H, C)$ and returns $\mathsf{unpad}_n(\mathcal{D}_r(sk_r, H, X))$ to $\mathcal{A}$. Finally, when $\mathcal{A}$ asks $\mathbf{PKout}()$, adversary $\mathcal{B}$ asks $pk_d \leftarrow \mathbf{PKout}()$, lets $\mathsf{pkout} = \mathsf{true}$, and returns $\langle pk_r, pk_d \rangle$ to $\mathcal{A}$.

We now turn to the second claim. Again, adversary $\mathcal{B}$ simulates $\mathcal{A}$ in its game instantiated with PKEAD. First, let $Q = \emptyset$ and execute $(pk_d, sk_d) \leftarrow\!\!\$\ \mathcal{K}_d(1^k)$. When $\mathcal{A}$ asks $\mathbf{LR}(H, M_0, M_1)$, adversary $\mathcal{B}$ asks $X \leftarrow\!\!\$\ \mathbf{LR}(H, M_0, M_1)$, computes the ciphertext, $C = \mathcal{E}_d(pk_d, \mathsf{pad}_n(X))$, adds (H, C) to Q, and returns C to $\mathcal{A}$. When $\mathcal{A}$ asks $\mathbf{Dec}(H, C)$, if $(H, C) \in Q$, then $\mathcal{B}$ returns $\lightning$; otherwise, it computes the intermediate value $X = \mathsf{unpad}_n(\mathcal{D}_d(sk_d, C))$, asks $M \leftarrow \mathbf{Dec}(H, X)$, and returns M to $\mathcal{A}$.

C.7 Theorem 9

First, by Theorem 2, there exists a (μ_1, v)-mm-adversary $\mathcal{A}_1$ that makes one $\mathbf{LR}$ query such that $\mathbf{Adv}^{\text{mm-cpa}}_{\mathsf{PKEAD},\mathcal{R}}(\mathcal{A}, k) \le q_e \cdot \mathbf{Adv}^{\text{mm-cpa}}_{\mathsf{PKEAD},\mathcal{R}}(\mathcal{A}_1, k)$. We bound $\mathcal{A}_1$'s advantage in the remainder.

Let $n = n(k)$, $\mu_1 = \mu_1(k)$, $\mu_2 = \mu_2(k)$, $v = v(k)$, $\rho = \rho(k)$, $\Pi = (\mathcal{K}_r, \mathcal{E}_r, \mathcal{D}_r)$, and F-DOAEP $= (\mathcal{K}_d, \mathcal{E}_d, \mathcal{D}_d)$. As usual, we assume that adversary $\mathcal{A}_1$'s $\mathbf{LR}$ query is not rejected. We first prove that PKEAD is MM-CPA secure. Following the proof of [BBO07, Theorem 5.2], we show the bound in the case that $n < k_0 + k_1$ where n is the length of the messages output by the source specified by $\mathcal{A}_1$ in its $\mathbf{LR}$ query. The OWF adversary $\mathcal{B}$ is defined in Figure 22. Note that $\mathcal{B}$ corresponds to the algorithm GetQuery specified in the proof of [BBO07, Theorem 5.2]. We bound $\mathcal{B}$'s advantage in the remainder.

The game $\mathbf{G}_0$ defined in Figure 19 by the code including the red statements and excluding the green statements. It is identical to $\mathbf{Exp}^{\text{mm-cpa}}_{\mathsf{PKEAD},\mathcal{R}}(\mathcal{A}_1, k)$. The game $\mathbf{G}_1$, which includes the green statements and excludes the red ones, is identical to $\mathbf{G}_0$ except for some code changes, which have no affect on the output of the oracles, but will assist in bounding the advantage $\mathcal{A}_1$ gets from its random oracle queries that precede its $\mathbf{LR}$ query.

Now refer to game $\mathbf{G}_2$ defined in Figure 20. This game is identical to $\mathbf{G}_1$ until bad_1 is set, which ocurrs *only if* adversary manages to ask both an H_1 query and an G query that coincide with its $\mathbf{LR}$ query *before* the $\mathbf{LR}$ query is made. More precisely, the probability that $\mathbf{G}_2(\mathcal{A}_1, k)$ sets bad_1 is at most the probability that $\mathcal{A}_1$ asked $a_1 = \mathsf{H}_1(A)$ and $\mathsf{G}(B)$, then asked $\mathbf{LR}(\boldsymbol{H}, \mathcal{M})$ where for some $i \in [v]$, it holds that $A = pk_d \,\|\, \boldsymbol{X}_r[i]$ and $B = pk_d \,\|\, (a_1 \oplus \boldsymbol{X}_\ell[i])$ where $X = \boldsymbol{X}_\ell[i] \,\|\, \boldsymbol{X}_r[i] = \mathcal{E}_r(pk_r, \boldsymbol{H}[i], \boldsymbol{M}_b[i]\,; \boldsymbol{r}[i])$. Because Π is injective, source $\mathcal{M}$ has min-entropy μ_1, and source $\mathcal{R}$ has min entropy μ_2, we have that $\Pr[\mathbf{G}_2(\mathcal{A}_1, k)\ \mathsf{sets}\ \mathsf{bad}_1] \le vq_1q_G/2^{\mu_1+\mu_2}$. By the fundamental lemma of game playing [BR06],

$$\Pr[\mathbf{G}_2(\mathcal{A}_1, k)] - \Pr[\mathbf{G}_0(\mathcal{A}_1, k)] \le \Pr[\mathbf{G}_2(\mathcal{A}_1, k)\ \mathsf{sets}\ \mathsf{bad}_1] \le \frac{vq_1q_G}{2^{\mu_1+\mu_2}}. \tag{40}$$

Next, refer to game $\mathbf{G}_3$ defined in Figure 20. This game is identical to $\mathbf{G}_2$ until bad_2 is set, which occurs only if the adversary manages to make a G query that coincides with its future $\mathbf{LR}$ query, but makes no such H_1 query. More precisely, the probability that $\mathbf{G}_3(\mathcal{A}_1, k)$ sets bad_2 is at most the probability that $\mathcal{A}_1$ asked $\mathsf{G}(B)$, then asked $\mathbf{LR}(\boldsymbol{H}, \mathcal{M})$ where for some $i \in [v]$, it holds that $B = pk_d \,\|\, (\boldsymbol{r}_1[i] \oplus \boldsymbol{X}_\ell[i])$ where $\boldsymbol{r}_1[i]$ is a uniform-random k_0-bit string and $X = \boldsymbol{X}_\ell[i] \,\|\, \boldsymbol{X}_r[i] = \mathcal{E}(pk_r, \boldsymbol{H}[i], \boldsymbol{M}_b[i]\,; \boldsymbol{r}[i])$. Since $\boldsymbol{r}_1[i]$ is uniform-random, so is $\boldsymbol{r}_1[i] \oplus \boldsymbol{X}_\ell[i]$ for any distribution on X. Hence,

$$\Pr[\mathbf{G}_3(\mathcal{A}_1, k)] - \Pr[\mathbf{G}_2(\mathcal{A}_1, k)] \le \Pr[\mathbf{G}_2(\mathcal{A}_1, k)\ \mathsf{sets}\ \mathsf{bad}_2] \le \frac{vq_G}{2^{k_0}}. \tag{41}$$

Next, refer to game $\mathbf{G}_4$ defined in Figure 21. This game is identical to $\mathbf{G}_3$ until $\mathsf{bad}_{3,1}$ is set or $\mathsf{bad}_{3,2}$ is set. First, observe that $\mathsf{bad}_{3,1}$ gets set only if $\mathsf{bad}_{3,2}$ gets set. Suppose that $\mathsf{bad}_{3,2} = \mathsf{true}$, but $\mathsf{bad}_{3,1} = \mathsf{false}$. This means that for every $i \in [v]$, both $\boldsymbol{r}_G[i]$ and $\boldsymbol{r}_2[i]$ are uniform-random strings from the perspective of the adversary. Since $\boldsymbol{S}_1[i] = \boldsymbol{r}_2[i] \oplus \boldsymbol{S}_0[i]$ and $\boldsymbol{T}_0[i] = \boldsymbol{r}_G[i] \oplus \boldsymbol{X}_r[i]$, the output of the $\mathbf{LR}$ oracle is identically distributed in both games. (Note that this is the step of the proof that uses the fact that DOAEP uses three Feistel rounds instead of the two used in OAEP.) It follows that the games are indistinguishable until $\mathsf{bad}_{3,1}$ gets set. Since $\boldsymbol{r}_G[i]$ is a uniform-random $(|X| - k_0)$-bit string, so is $\boldsymbol{T}_0[i] = \boldsymbol{r}_G[i] \oplus \boldsymbol{X}_r[i]$. We conclude that

$$\Pr[\mathbf{G}_4(\mathcal{A}_1, k)] - \Pr[\mathbf{G}_3(\mathcal{A}_1, k)] \le \Pr[\mathbf{G}_4(\mathcal{A}_1, k)\ \mathsf{sets}\ \mathsf{bad}_{3,1}] \le \frac{vq_2}{2^{k_0}}. \tag{42}$$

Games $\mathbf{G}_5$ and $\mathbf{G}_4$ defined in Figure 21 are identical until bad_4 gets set, which occurs only if the adversary asks an H_2 query that coincides with its future $\mathbf{LR}$ query, but it makes no such H_1 or G query. Hence,

$$\Pr[\mathbf{G}_5(\mathcal{A}_1, k)] - \Pr[\mathbf{G}_4(\mathcal{A}_1, k)] \le \Pr[\mathbf{G}_5(\mathcal{A}_1, k)\ \mathsf{sets}\ \mathsf{bad}_4] \le \frac{vq_2}{2^{k_0}}. \tag{43}$$

At this point, we have bounded the advantage the adversary gets from its random oracles queries before making its $\mathbf{LR}$ query. Finally, refer to game $\mathbf{G}_6$ defined in Figure 22. Game $\mathbf{G}_5$ corresponds to G_1 in the proof of [BBO07, Theorem 5.2] and $\mathbf{G}_6$ to corresponds to G_8. Applying the same argument justifying equations (10) through (20) of [BBO07] yields the claim.

We now prove that PKEAD is IND-CPA secure. On input $(1^k, pk)$ and oracle $\mathbf{LR}$, adversary $\mathcal{B}$ does as follows: run $(f, f^{-1}) \leftarrow\!\$\ F(1^k)$. Run $\mathcal{A}_1$ on input $(1^k, \langle pk, \langle f\rangle\rangle)$ and oracle $\mathbf{LR}'$, defined as follows: when $\mathcal{A}_1$ asks $\mathbf{LR}'(H, M_0, M_1)$, ask (H, M_0, M_1) of $\mathbf{LR}$, getting X in response. Compute $C = \mathcal{E}_d(\langle f\rangle, \mathsf{pad}_n(X))$ and return C to $\mathcal{A}_1$. Finally, output whatever $\mathcal{A}_1$ outputs. Since $\mathcal{B}$ perfectly simulates the IND-CPA game instantiated with PKEAD, we have that $\mathbf{Adv}^{\text{ind-cpa}}_{\Pi}(\mathcal{B}, k) = \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{PKEAD}}(\mathcal{A}_1, k)$.

D Games for proofs

```
$\mathbf{H}_i(\mathcal{A})$ $\mathbf{H}'_i(\mathcal{A})$
$Q \leftarrow \emptyset;\ j \leftarrow 0$
$(pk_0, sk_0) \leftarrow\$ \mathsf{Kgen}(1^k)$
$(pk_1, sk_1) \leftarrow\$ \mathsf{Kgen}(1^k)$
$a \leftarrow\$ \mathcal{A}^{\mathbf{LR},\mathbf{Dec},\mathbf{PKout}}(1^k)$
return $a$
```

```
Oracle $\mathbf{LR}(\boldsymbol{H}, \mathcal{M})$
if $\mathsf{pkout} = \mathsf{true}$ then return $\lightning$
$j \leftarrow j + 1$
if $j > q - i$ then $b \leftarrow 0$ else $b \leftarrow 1$
$d \leftarrow 0$; if $j = i$ then $d \leftarrow 1$
$(\boldsymbol{M}_0, \boldsymbol{M}_1, \boldsymbol{r}) \leftarrow\$ \mathcal{M}(1^k)$
$\boldsymbol{C} \leftarrow \mathsf{Enc}(pk_d, \boldsymbol{H}, \boldsymbol{M}_b\,; \boldsymbol{r})$
for $i \leftarrow 1$ to $|\boldsymbol{H}|$ do
    $Q \leftarrow Q \cup \{(\boldsymbol{H}_i, \boldsymbol{C}_i)\}$
return $\boldsymbol{C}$
```

```
Oracle $\mathbf{Dec}(H, C)$
if $(H, C) \in Q$ then return $\lightning$
return $\mathsf{Dec}(sk_0, H, C)$

Oracle $\mathbf{PKout}()$
$\mathsf{pkout} \leftarrow \mathsf{true}$; return $pk_0$
```

```
$\mathcal{D}_i^{\mathcal{A},\mathbf{LR},\mathbf{Enc},\mathbf{Dec}}(1^k)$ $\overline{\mathcal{D}}_i^{\mathcal{A},\mathbf{LR},\mathbf{Enc},\mathbf{Dec}}(1^k)$
$Q \leftarrow \emptyset;\ j \leftarrow 0$
$a \leftarrow\$ \mathcal{A}^{\mathbf{LR}',\mathbf{Dec}',\mathbf{PKout}'}(1^k)$
$g \leftarrow a \oplus 1$; $g \leftarrow a$
return $g$

Oracle $\mathbf{LR}'(\boldsymbol{H}, \mathcal{M})$
$j \leftarrow j + 1$
if $j < i$ then $\boldsymbol{C} \leftarrow\$ \mathbf{Enc}(\boldsymbol{H}, \mathcal{M}_1)$
else if $j = i$ then
    $(pk_0, pk_1, \boldsymbol{C}) \leftarrow\$ \mathbf{LR}(\boldsymbol{H}, \mathcal{M}_1)$
else
    $(\boldsymbol{M}_0, \boldsymbol{M}_1, \boldsymbol{r}) \leftarrow\$ \mathcal{M}(1^k)$
    $\boldsymbol{C} \leftarrow \mathsf{Enc}(pk_0, \boldsymbol{H}, \boldsymbol{M}_0\,; \boldsymbol{r})$
for $i \leftarrow 1$ to $|\boldsymbol{H}|$ do $Q \leftarrow Q \cup \{(\boldsymbol{H}_i, \boldsymbol{C}_i)\}$
return $\boldsymbol{C}$

$\mathcal{M}_b(1^k)$  # An mr-source
$(\boldsymbol{M}_0, \boldsymbol{M}_1, \boldsymbol{r}) \leftarrow\$ \mathcal{M}(1^k)$; return $(\boldsymbol{M}_b, \boldsymbol{r})$

Oracle $\mathbf{Dec}'(H, C)$
if $(H, C) \in Q$ then return $\lightning$
return $\mathbf{Dec}_0(H, C)$

Oracle $\mathbf{PKout}'()$
return $pk_0$
```

```
$\mathcal{B}_i^{\mathcal{A},\mathbf{LR},\mathbf{Dec},\mathbf{PKout}}(1^k)$
$Q \leftarrow \emptyset;\ j \leftarrow 0$
$(pk_0, sk_0) \leftarrow\$ \mathsf{Kgen}(1^k)$
$a \leftarrow\$ \mathcal{A}^{\mathbf{LR}',\mathbf{Dec}',\mathbf{PKout}}(1^k)$
return $a$

Oracle $\mathbf{LR}'(\boldsymbol{H}, \mathcal{M})$
$j \leftarrow j + 1$; $(\boldsymbol{M}_0, \boldsymbol{M}_1, \boldsymbol{r}) \leftarrow\$ \mathcal{M}(1^k)$
if $j < i$ then $\boldsymbol{C} \leftarrow \mathsf{Enc}(pk_0, \boldsymbol{H}, \boldsymbol{M}_1\,; \boldsymbol{r})$
if $j = i$ then $\boldsymbol{C} \leftarrow\$ \mathbf{LR}(\boldsymbol{H}, \mathcal{M})$
else $\boldsymbol{C} \leftarrow \mathsf{Enc}(pk_0, \boldsymbol{H}, \boldsymbol{M}_0\,; \boldsymbol{r})$
for $i \leftarrow 1$ to $|\boldsymbol{H}|$ do $Q \leftarrow Q \cup \{(\boldsymbol{H}_i, \boldsymbol{C}_i)\}$
return $\boldsymbol{C}$

Oracle $\mathbf{Dec}'(H, C)$
if $(H, C) \in Q$ then return $\lightning$
return $\mathbf{Dec}(H, C)$
```

Fig. 9. Game and adversaries for proof of Theorem 1.

```
$\underline{\mathbf{H}_b(\mathcal{A}')}$  $\mathbf{H}'_i(\mathcal{A}')$
$Q \leftarrow \emptyset$
$(pk_0, sk_0) \leftarrow\!\!\$\ \mathsf{Kgen}(1^k)$
$(pk_1, sk_1) \leftarrow\!\!\$\ \mathsf{Kgen}(1^k)$
$a \leftarrow\!\!\$\ \mathcal{A}'^{\mathbf{LR},\mathbf{Dec},\mathbf{PKout}}(1^k, pk_0)$
return $a$
```

```
$\underline{\textbf{Oracle LR}(\boldsymbol{H}, \mathcal{M})}$
$\boldsymbol{r} \leftarrow\!\!\$\ \mathcal{R}(1^k)$; $(\boldsymbol{M}_0, \boldsymbol{M}_1) \leftarrow\!\!\$\ \mathcal{M}(1^k)$
$d \leftarrow 0$;  $d \leftarrow 1$
$\boldsymbol{C} \leftarrow \mathsf{Enc}(pk_d, \boldsymbol{H}, \boldsymbol{M}_b\, ; \boldsymbol{r})$
for $i \leftarrow 1$ to $|\boldsymbol{H}|$ do
  $Q \leftarrow Q \cup \{(\boldsymbol{H}_i, \boldsymbol{C}_i)\}$
return $\boldsymbol{C}$
```

```
$\underline{\textbf{Oracle Dec}(H, C)}$
if $(H, C) \in Q$ then return $\lightning$
return $\mathsf{Dec}(sk_0, H, C)$

$\underline{\textbf{Oracle PKout}()}$
$\mathsf{pkout} \leftarrow \mathsf{true}$; return $pk_1$
```

```
$\underline{\mathcal{D}_b^{\mathcal{A}',\mathbf{LR},\mathbf{Enc},\mathbf{Dec}}(1^k)}$
$x, y \leftarrow\!\!\$\ \{0,1\}$; $(pk', sk') \leftarrow\!\!\$\ \mathsf{Kgen}(1^k)$
$a \leftarrow\!\!\$\ \mathcal{A}'^{\mathbf{LR}',\mathbf{Dec}',\mathbf{PKout}'}(1^k, pk')$
return $(\neg a) \oplus y \oplus b$

$\underline{\textbf{Oracle LR}'(\boldsymbol{H}, \mathcal{M})}$
if $x = 1$ then $(pk_0, pk_1, \boldsymbol{C}) \leftarrow\!\!\$\ \mathbf{LR}(\boldsymbol{H}, \mathcal{M}_b)$
else
  $\boldsymbol{r} \leftarrow\!\!\$\ \mathcal{R}(1^k)$; $\boldsymbol{M}_b \leftarrow\!\!\$\ \mathcal{M}_b(1^k)$
  $\boldsymbol{C} \leftarrow \mathsf{Enc}(pk', \boldsymbol{H}, \boldsymbol{M}_b;\ \boldsymbol{r})$
for $i \leftarrow 1$ to $|\boldsymbol{H}|$ do $Q \leftarrow Q \cup \{(\boldsymbol{H}_i, \boldsymbol{C}_i)\}$
return $\boldsymbol{C}$

$\underline{\mathcal{M}_b(1^k)}$
$(\boldsymbol{M}_0, \boldsymbol{M}_1) \leftarrow\!\!\$\ \mathcal{M}(1^k)$; return $\boldsymbol{M}_b$

$\underline{\textbf{Oracle Dec}'(H, C)}$
if $(H, C) \in Q$ then return $\lightning$
if $x = 1$ then return $\mathbf{Dec}_y(H, C)$
else return $\mathsf{Dec}(sk', H, C)$

$\underline{\textbf{Oracle PKout}'()}$
if $x = 0$ then $(pk_0, pk_1, \Lambda) \leftarrow \mathbf{LR}(\bot, \bot)$
return $pk_y$
```

```
$\underline{\mathcal{B}^{\mathcal{A}',\mathbf{LR},\mathbf{Dec},\mathbf{PKout}}(1^k)}$
$(pk', sk') \leftarrow\!\!\$\ \mathsf{Kgen}(1^k)$
$a \leftarrow\!\!\$\ \mathcal{A}'^{\mathbf{LR}',\mathbf{Dec}',\mathbf{PKout}'}(1^k, pk')$
return $a$

$\underline{\textbf{Oracle LR}'(\boldsymbol{H}, \mathcal{M})}$
$\boldsymbol{C} \leftarrow\!\!\$\ \mathbf{LR}(\boldsymbol{H}, \mathcal{M}')$
for $i \leftarrow 1$ to $|\boldsymbol{H}|$ do $Q \leftarrow Q \cup \{(\boldsymbol{H}_i, \boldsymbol{C}_i)\}$
return $\boldsymbol{C}$

$\underline{\mathcal{M}'(1^k)}$
$\boldsymbol{r} \leftarrow\!\!\$\ \mathcal{R}(1^k)$; $(\boldsymbol{M}_0, \boldsymbol{M}_1) \leftarrow\!\!\$\ \mathcal{M}(1^k)$
return $(\boldsymbol{M}_0, \boldsymbol{M}_1, \boldsymbol{r})$

$\underline{\textbf{Oracle Dec}'(H, C)}$
if $(H, C) \in Q$ then return $\lightning$
return $\mathbf{Dec}(H, C)$

$\underline{\textbf{Oracle PKout}'()}$
return $\mathbf{PKout}()$
```

Fig. 10. Game and adversaries for proof of Theorem 3.

```
$\mathbf{G}_0(\mathcal{A}_1, k)$:
$Q, Q_1, Q_G, Q_2 \leftarrow \emptyset$
$(pk, sk) \xleftarrow{\$} \mathsf{Kgen}(1^k)$; $\boldsymbol{r} \xleftarrow{\$} \mathcal{R}^{\mathsf{H}_1,\mathsf{G},\mathsf{H}_2}(1^k)$
for $i \leftarrow 1$ to $v$ do $A_i \xleftarrow{\$} \{0,1\}^{n-\rho-8}$
$b \xleftarrow{\$} \{0,1\}$; $b' \xleftarrow{\$} \mathcal{A}_1{}^{\mathbf{LR},\mathbf{Dec},\mathsf{H}_1,\mathsf{G},\mathsf{H}_2}(1^k, pk)$
return $b = b'$

Oracle $\mathbf{LR}(\boldsymbol{H}, \mathcal{M})$:
$\boldsymbol{r} \xleftarrow{\$} \mathcal{R}^{\mathsf{H}_1,\mathsf{G},\mathsf{H}_2}(1^k)$
$(\boldsymbol{M}_0, \boldsymbol{M}_1) \xleftarrow{\$} \mathcal{M}^{\mathsf{H}_1,\mathsf{G},\mathsf{H}_2}(1^k)$
$\langle f \rangle \leftarrow pk$
for $i \leftarrow 1$ to $v$ do
    $PM \leftarrow \mathsf{pad}(\boldsymbol{M}_{b,i})$
    $X_0 \leftarrow PM \,\|\, \mathsf{H}_1(\boldsymbol{H}_i)$; $Y_0 \leftarrow \boldsymbol{r}_i$
    $X_1 \leftarrow X_0 \oplus$ $A_i$ $\mathsf{G}(Y_0)$
    $Y_1 \leftarrow Y_0 \oplus \mathsf{H}_2(X_1)$
    $P \leftarrow X_1 \,\|\, Y_1 \,\|\, [0]$; $\boldsymbol{C}_i \leftarrow f(P)$
for $i \leftarrow 1$ to $v$ do $Q \leftarrow Q \cup \{(\boldsymbol{H}_i, \boldsymbol{C}_i)\}$
return $\boldsymbol{C}$

Oracle $\mathsf{H}_1(X)$
if $T_1[X] = \bot$ then $T_1[X] \xleftarrow{\$} \{0,1\}^{\tau}$
$Q_1 \leftarrow Q_1 \cup \{X\}$; return $T_1[X]$
```

```
$\mathbf{G}_1(\mathcal{A}_1, k)$

Oracle $\mathbf{Dec}(H, C)$:
if $(H, C) \in Q$ then return $\lightning$
$\langle f^{-1} \rangle \leftarrow sk$; $P \leftarrow f^{-1}(C)$
if $|P| \neq n$ then return $\bot$
$X_1 \,\|\, Y_1 \,\|\, [z] \leftarrow P$  # $|Y_1| = \rho$
$Y_0 \leftarrow Y_1 \oplus \mathsf{H}_2(X_1)$
$X_0 \leftarrow X_1 \oplus \mathsf{G}(Y_0)$
$PM \,\|\, T \leftarrow X_0$ # where $|T| = \tau$
$T^* \leftarrow \mathsf{H}_1(H)$
if $T^* \neq T$ then return $\bot$
return $\mathsf{unpad}(PM)$

Oracle $\mathsf{G}(X)$
for $i \leftarrow 1$ to $v$ do
    if $X = \boldsymbol{r}_i$ then $\mathsf{bad}_1 \leftarrow \mathsf{true}$; return $A_i$
if $T_G[X] = \bot$ then $T_G[X] \xleftarrow{\$} \{0,1\}^m$
$Q_G \leftarrow Q_G \cup \{X\}$; return $T_G[X]$

Oracle $\mathsf{H}_2(X)$
if $T_2[X] = \bot$ then $T_2[X] \xleftarrow{\$} \{0,1\}^{\rho}$
$Q_2 \leftarrow Q_2 \cup \{X\}$; return $T_2[X]$
```

```
Oracle $\mathbf{LR}(\boldsymbol{H}, \mathcal{M})$:                $\mathbf{G}_2(\mathcal{A}_1, k)$
$(\boldsymbol{M}_0, \boldsymbol{M}_1) \xleftarrow{\$} \mathcal{M}^{\mathsf{H}_1,\mathsf{G},\mathsf{H}_2}(1^k)$; $\langle f \rangle \leftarrow pk$
for $i \leftarrow 1$ to $v$ do
    $PM \leftarrow \mathsf{pad}(\boldsymbol{M}_{b,i})$
    $X_0 \leftarrow PM \,\|\, \mathsf{H}_1(\boldsymbol{H}_i)$; $Y_0 \leftarrow \boldsymbol{r}_i$
    $X_1 \leftarrow$ $X_0 \oplus$ $A_i$
    $Y_1 \leftarrow Y_0 \oplus \mathsf{H}_2(X_1)$
    $P \leftarrow X_1 \,\|\, Y_1 \,\|\, [0]$; $\boldsymbol{C}_i \leftarrow f(P)$
for $i \leftarrow 1$ to $v$ do $Q \leftarrow Q \cup \{(\boldsymbol{H}_i, \boldsymbol{C}_i)\}$
return $\boldsymbol{C}$

Oracle $\mathsf{G}(X)$
for $i \leftarrow 1$ to $v$ do
    if $X = \boldsymbol{r}_i$ then $\mathsf{bad}_1 \leftarrow \mathsf{true}$; return $A_i$
if $T_G[X] = \bot$ then $T_G[X] \xleftarrow{\$} \{0,1\}^m$
$Q_G \leftarrow Q_G \cup \{X\}$; return $T_G[X]$
```

```
Oracle $\mathbf{LR}(\boldsymbol{H}, \mathcal{M})$:                $\mathbf{G}_4(\mathcal{A}_1, k)$
$(\boldsymbol{M}_0, \boldsymbol{M}_1) \xleftarrow{\$} \mathcal{M}^{\mathsf{H}_1,\mathsf{G},\mathsf{H}_2}(1^k)$; $\langle f \rangle \leftarrow pk$
for $i \leftarrow 1$ to $v$ do
    $PM \leftarrow \mathsf{pad}(\boldsymbol{M}_{b,i})$
    $X_0 \leftarrow PM \,\|\, \mathsf{H}_1(\boldsymbol{H}_i)$; $Y_0 \leftarrow \boldsymbol{r}_i$
    $X_1 \leftarrow A_i$
    $Y_1 \leftarrow$ $Y_0 \oplus$ $B_i$
    $P \leftarrow X_1 \,\|\, Y_1 \,\|\, [0]$; $\boldsymbol{C}_i \leftarrow f(P)$
for $i \leftarrow 1$ to $v$ do $Q \leftarrow Q \cup \{(\boldsymbol{H}_i, \boldsymbol{C}_i)\}$
return $\boldsymbol{C}$

Oracle $\mathsf{H}_2(X)$
for $i \leftarrow 1$ to $v$ do
    if $X = A_i$ then $\mathsf{bad}^* \leftarrow \mathsf{true}$; return $B_i$
if $T_2[X] = \bot$ then $T_2[X] \xleftarrow{\$} \{0,1\}^{\rho}$
$Q_2 \leftarrow Q_2 \cup \{X\}$; return $T_2[X]$
```

Fig. 11. Games 0, 1, 2, and 4 for proof of Theorem 5.

```
$\mathbf{G}_2(\mathcal{A}_1, k)$:
$Q, Q_1, Q_G, Q_2 \leftarrow \emptyset$
$(pk, sk) \leftarrow\$ \mathsf{Kgen}(1^k)$; $\boldsymbol{r} \leftarrow\$ \mathcal{R}^{\mathsf{H}_1,\mathsf{G},\mathsf{H}_2}(1^k)$
for $i \leftarrow 1$ to $v$ do
  $A_i \leftarrow\$ \{0,1\}^{n-\rho-8}$; $B_i \leftarrow\$ \{0,1\}^{\rho}$
$b \leftarrow\$ \{0,1\}$; $b' \leftarrow\$ \mathcal{A}_1^{\mathbf{LR},\mathbf{Dec},\mathsf{H}_1,\mathsf{G},\mathsf{H}_2}(1^k, pk)$
return $b = b'$

Oracle $\mathbf{LR}(\boldsymbol{H}, \mathcal{M})$:
$(\boldsymbol{M}_0, \boldsymbol{M}_1) \leftarrow\$ \mathcal{M}^{\mathsf{H}_1,\mathsf{G},\mathsf{H}_2}(1^k)$; $\langle f \rangle \leftarrow pk$
for $i \leftarrow 1$ to $v$ do
  $PM \leftarrow \mathsf{pad}(\boldsymbol{M}_{b,i})$
  $X_0 \leftarrow PM \| \mathsf{H}_1(\boldsymbol{H}_i)$; $Y_0 \leftarrow \boldsymbol{r}_i$
  $X_1 \leftarrow A_i$
  $Y_1 \leftarrow Y_0 \oplus$ $B_i$ $\mathsf{H}_2(X_1)$
  $P \leftarrow X_1 \| Y_1 \| [0]$; $\boldsymbol{C}_i \leftarrow f(P)$
for $i \leftarrow 1$ to $v$ do $Q \leftarrow Q \cup \{(\boldsymbol{H}_i, \boldsymbol{C}_i)\}$
return $\boldsymbol{C}$

Oracle $\mathsf{H}_1(X)$
if $T_1[X] = \perp$ then $T_1[X] \leftarrow\$ \{0,1\}^{\tau}$
$Q_1 \leftarrow Q_1 \cup \{X\}$; return $T_1[X]$
```

```
Oracle $\mathbf{Dec}(H, C)$:                                  $\mathbf{G}_3(\mathcal{A}_1, k)$
if $(H, C) \in Q$ then return $\lightning$
$\langle f^{-1} \rangle \leftarrow sk$; $P \leftarrow f^{-1}(C)$
if $|P| \neq n$ then return $\perp$
$X_1 \| Y_1 \| [z] \leftarrow P$  # $|Y_1| = \rho$
$Y_0 \leftarrow Y_1 \oplus \mathsf{H}_2(X_1)$
$X_0 \leftarrow X_1 \oplus \mathsf{G}(Y_0)$
$PM \| T \leftarrow X_0$ # where $|T| = \tau$
$T^* \leftarrow \mathsf{H}_1(H)$
if $T^* \neq T$ then return $\perp$
return $\mathsf{unpad}(PM)$

Oracle $\mathsf{G}(X)$
for $i \leftarrow 1$ to $v$ do
  if $X = \boldsymbol{r}_i$ then $\mathsf{bad}_1 \leftarrow \mathsf{true}$
if $T_G[X] = \perp$ then $T_G[X] \leftarrow\$ \{0,1\}^m$
$Q_G \leftarrow Q_G \cup \{X\}$; return $T_G[X]$

Oracle $\mathsf{H}_2(X)$
for $i \leftarrow 1$ to $v$ do
  if $X = A_i$ then $\mathsf{bad}^* \leftarrow \mathsf{true}$; return $B_i$
if $T_2[X] = \perp$ then $T_2[X] \leftarrow\$ \{0,1\}^{\rho}$
$Q_2 \leftarrow Q_2 \cup \{X\}$; return $T_2[X]$
```

```
$\mathbf{G}_4(\mathcal{A}_1, k)$:
$Q, Q_1, Q_G, Q_2 \leftarrow \emptyset$
$(pk, sk) \leftarrow\$ \mathsf{Kgen}(1^k)$; $\boldsymbol{r} \leftarrow\$ \mathcal{R}^{\mathsf{H}_1,\mathsf{G},\mathsf{H}_2}(1^k)$
for $i \leftarrow 1$ to $v$ do
  $A_i \leftarrow\$ \{0,1\}^{n-\rho-8}$; $B_i \leftarrow\$ \{0,1\}^{\rho}$
$b \leftarrow\$ \{0,1\}$; $b' \leftarrow\$ \mathcal{A}_1^{\mathbf{LR},\mathbf{Dec},\mathsf{H}_1,\mathsf{G},\mathsf{H}_2}(1^k, pk)$
return $b = b'$

Oracle $\mathbf{LR}(\boldsymbol{H}, \mathcal{M})$:
$(\boldsymbol{M}_0, \boldsymbol{M}_1) \leftarrow\$ \mathcal{M}^{\mathsf{H}_1,\mathsf{G},\mathsf{H}_2}(1^k)$ ; $\langle f \rangle \leftarrow pk$
for $i \leftarrow 1$ to $v$ do
  $PM \leftarrow \mathsf{pad}(\boldsymbol{M}_{b,i})$
  $X_0 \leftarrow PM \| \mathsf{H}_1(\boldsymbol{H}_i)$; $Y_0 \leftarrow \boldsymbol{r}_i$
  $X_1 \leftarrow A_i$; $Y_1 \leftarrow B_i$
  $P \leftarrow X_1 \| Y_1 \| [0]$; $\boldsymbol{C}_i \leftarrow f(P)$
for $i \leftarrow 1$ to $v$ do $Q \leftarrow Q \cup \{(\boldsymbol{H}_i, \boldsymbol{C}_i)\}$
return $\boldsymbol{C}$
```

```
Oracle $\mathbf{Dec}(H, C)$:                                  $\mathbf{G}_5(\mathcal{A}_1, k)$
if $(H, C) \in Q$ then return $\lightning$
$\langle f^{-1} \rangle \leftarrow sk$; $P \leftarrow f^{-1}(C)$
if $|P| \neq n$ then return $\perp$
$X_1 \| Y_1 \| [z] \leftarrow P$  # $|Y_1| = \rho$
if $T_2[X_1]$ undefined then $\mathsf{bad}_6' \leftarrow \mathsf{true}$
$Y_0 \leftarrow Y_1 \oplus \mathsf{H}_2(X_1)$
if $T_G[Y_0]$ undefined then $\mathsf{bad}_5' \leftarrow \mathsf{true}$
$X_0 \leftarrow X_1 \oplus \mathsf{G}(Y_0)$
$PM \| T \leftarrow X_0$ # where $|T| = \tau$
$T^* \leftarrow \mathsf{H}_1(H)$
if $T^* \neq T$ then return $\perp$
if $(\forall H^* \in Q_1)\, T_1[H^*] \neq T$ then return $\perp$
else if $\mathsf{bad}_5'$ then $\mathsf{bad}_5 \leftarrow \mathsf{true}$
else if $\mathsf{bad}_6'$ then $\mathsf{bad}_6 \leftarrow \mathsf{true}$
return $\mathsf{unpad}(PM)$
```

Fig. 12. Games 3 and 5 for proof of Theorem 5.

```
Oracle Dec(H, C):                                        G_6(A_1, k)
if $(H,C) \in Q$ then return $\lightning$
$\langle f^{-1}\rangle \leftarrow sk$; $P \leftarrow f^{-1}(C)$
if $|P| \neq n$ then return $\perp$
$X_1 \| Y_1 \| [z] \leftarrow P$  # $|Y_1| = \rho$
if $T_2[X_1]$ undefined then $\mathsf{bad}'_6 \leftarrow \mathsf{true}$
$Y_0 \leftarrow Y_1 \oplus \mathsf{H}_2(X_1)$
if $T_G[Y_0]$ undefined then $\mathsf{bad}'_5 \leftarrow \mathsf{true}$
$X_0 \leftarrow X_1 \oplus \mathsf{G}(Y_0)$
$PM \| T \leftarrow X_0$ # where $|T| = \tau$
$T^* \leftarrow \mathsf{H}_1(H)$
if $(\forall H^* \in Q_1)\, T_1[H^*] \neq T$ then return $\perp$
else if $\mathsf{bad}'_5$ then $\mathsf{bad}_5 \leftarrow \mathsf{true}$; return $\perp$
else if $\mathsf{bad}'_6$ then $\mathsf{bad}_6 \leftarrow \mathsf{true}$
return $\mathsf{unpad}(PM)$
```

```
Oracle Dec(H, C):                                        G_7(A_1, k)
if $(H,C) \in Q$ then return $\lightning$
$\langle f^{-1}\rangle \leftarrow sk$; $P \leftarrow f^{-1}(C)$
if $|P| \neq n$ then return $\perp$
$X_1 \| Y_1 \| [z] \leftarrow P$  # $|Y_1| = \rho$
if $T_2[X_1]$ undefined then $\mathsf{bad}'_6 \leftarrow \mathsf{true}$
$Y_0 \leftarrow Y_1 \oplus \mathsf{H}_2(X_1)$
if $T_G[Y_0]$ undefined then $\mathsf{bad}'_5 \leftarrow \mathsf{true}$
$X_0 \leftarrow X_1 \oplus \mathsf{G}(Y_0)$
$PM \| T \leftarrow X_0$ # where $|T| = \tau$
$T^* \leftarrow \mathsf{H}_1(H)$
if $(\forall H^* \in Q_1)\, T_1[H^*] \neq T$ then return $\perp$
else if $\mathsf{bad}'_5$ then $\mathsf{bad}_5 \leftarrow \mathsf{true}$; return $\perp$
else if $\mathsf{bad}'_6$ then $\mathsf{bad}_6 \leftarrow \mathsf{true}$; return $\perp$
return $\mathsf{unpad}(PM)$
```

```
$\mathcal{B}(1^k, f, y)$:
$Q, Q_1 Q_G, Q_2 \leftarrow \emptyset$
$pk \leftarrow \langle f\rangle$; $j \leftarrow\$ [v]$
$b' \leftarrow\$ \mathcal{A}_1^{\mathbf{LR}', \mathbf{Dec}', \mathsf{H}_1, \mathsf{G}, \mathsf{H}_2}(1^k, pk)$
$x' \leftarrow\$ Q_2$; return $x'$

Oracle LR'(H, M):
for $i \leftarrow 1$ to $v$ do
  if $i = j$ then $\boldsymbol{C}_i \leftarrow y$
  else $P \leftarrow\$ \{0,1\}^{n-8} \| [0]$; $\boldsymbol{C}_i \leftarrow f(P)$
for $i \leftarrow 1$ to $v$ do $Q \leftarrow Q \cup \{(\boldsymbol{H}_i, \boldsymbol{C}_i)\}$
return $\boldsymbol{C}$

Oracle H_1(X)
if $T_1[X] = \perp$ then $T_1[X] \leftarrow\$ \{0,1\}^\tau$
$Q_1 \leftarrow Q_1 \cup \{X\}$; return $T_1[X]$
```

```
Oracle Dec'(H, C)
if $(H,C) \in Q$ then return $\lightning$
for each $A \in Q_G$ do
  for each $B \in Q_2$ do
    $U \leftarrow \mathsf{G}(A) \oplus B$; $V \leftarrow \mathsf{H}_2(B) \oplus A$
    $PM \| T \leftarrow U$  # where $|T| = \tau$
    if $f(B \| V \| [0]) = C$ then
      for each $H^* \in Q_1$ do
        if $T_1[H^*] = T$ then
          return $\mathsf{unpad}(PM)$
return $\perp$

Oracle G(X)
if $T_G[X] = \perp$ then $T_G[X] \leftarrow\$ \{0,1\}^m$
$Q_G \leftarrow Q_G \cup \{X\}$; return $T_G[X]$

Oracle H_2(X)
if $T_2[X] = \perp$ then $T_2[X] \leftarrow\$ \{0,1\}^\rho$
$Q_2 \leftarrow Q_2 \cup \{X\}$; return $T_2[X]$
```

Fig. 13. Games 6 and 7 and adversary $\mathcal{B}$ for proof of Theorem 5.

```
$\mathbf{Experiment}\ \mathbf{G}_0(\mathcal{A},k)$:
$Q, AD \leftarrow \emptyset$; $\mathsf{pkout} \leftarrow \mathsf{false}$
$R \leftarrow_{\$} \{0,1\}^r$
$(f, f^{-1} \leftarrow_{\$} F(1^k))$
$pk \leftarrow f \,\|\, R$ ; $sk \leftarrow f^{-1}$
$b \leftarrow_{\$} \{0,1\}$
$b' \leftarrow_{\$} \mathcal{A}^{\mathbf{LR},\mathbf{Dec},\mathbf{PKout},\mathsf{H}_1,\mathsf{H}_2}(1^k)$
return $b = b'$

$\mathbf{Oracle\ LR}(\boldsymbol{H}, \mathcal{M})$:
if $\mathsf{pkout} = \mathsf{true}$ then return $\lightning$
$(\boldsymbol{M}_0, \boldsymbol{M}_1, \boldsymbol{r}) \leftarrow_{\$} \mathcal{M}^{\mathsf{H}_1,\mathsf{H}_2}(1^k)$
For $i = 1$ to $v(k)$
    $K_P \leftarrow \mathsf{H}_1(\langle f \,\|\, R, \boldsymbol{H}[i], \boldsymbol{M}_b[i], \boldsymbol{r}[i]\rangle)$
    $C_1 \leftarrow f(K_P)$
    $K \leftarrow \mathsf{H}_2(\langle f \,\|\, R, \boldsymbol{H}[i], K_P\rangle)$
    $\tilde{H} \leftarrow \langle \boldsymbol{H}[i], C_1\rangle$; $AD \leftarrow AD \cup \tilde{H}$
    $IV \leftarrow \mathsf{extract}(\tilde{H})$
    $C_2 \leftarrow \mathsf{AEAD.Enc}(K, IV, \tilde{H}, \boldsymbol{M}_b[i])$
    $\boldsymbol{C}[i] \leftarrow C_1 \,\|\, C_2$
for $i \leftarrow 1$ to $|\boldsymbol{H}|$ do $Q \leftarrow Q \cup \{(\boldsymbol{H}_i, \boldsymbol{C}_i)\}$
return $\boldsymbol{C}$

$\mathbf{Oracle\ Dec}(H, C)$:
if $(H, C) \in Q$ then return $\lightning$
return $\mathsf{HE.Dec}_{sk}(H^*, C_1^* \,\|\, C_2^*)$

$\mathbf{Oracle\ PKout}()$:
$\mathsf{pkout} \leftarrow \mathsf{true}$; return $pk$

$\mathbf{Oracle}\ \mathsf{H}_1(\langle P, H, M, X\rangle)$:
$Y \leftarrow_{\$} \{0,1\}^{k_P}$
If $T_1[\langle P, H, M, X\rangle] = \perp$
then $T_1[\langle P, H, M, X\rangle] \leftarrow Y$
return $T_1[\langle P, H, M, X\rangle]$

$\mathbf{Oracle}\ \mathsf{H}_2(\langle P, H, KP\rangle)$:
$Y \leftarrow_{\$} \{0,1\}^{\lambda}$
If $T_2[\langle P, H, KP\rangle] = \perp$
then $T_2[\langle P, H, KP\rangle] \leftarrow Y$
return $T_2[\langle P, H, KP\rangle]$
```

```
$\mathbf{Experiment}\ \mathbf{G}_1(\mathcal{A},k)$:
Is like $\mathbf{G}_0$ except:
$\mathbf{Oracle}\ \mathsf{H}_1(\langle P, H, M, X\rangle)$:
$Y \leftarrow_{\$} \{0,1\}^{k_P}$
If $P = pk$ and $T_1[\langle P, H, M, X\rangle] \neq \perp$
then $\mathsf{bad}_1 \leftarrow true$
If $T_1[\langle P, H, M, X\rangle] = \perp$
then $T_1[\langle P, H, M, X\rangle] \leftarrow Y$
return $T_1[\langle P, H, M, X\rangle]$

$\mathbf{Experiment}\ \mathbf{G}_2(\mathcal{A},k)$:
Is like $\mathbf{G}_1$ except:
$\mathbf{Oracle\ LR}(\boldsymbol{H}, \mathcal{M})$:
if $\mathsf{pkout} = \mathsf{true}$ then return $\lightning$
$(\boldsymbol{M}_0, \boldsymbol{M}_1, \boldsymbol{r}) \leftarrow_{\$} \mathcal{M}^{\mathsf{H}_1,\mathsf{H}_2}(1^k)$
For $i = 1$ to $v(k)$
    $K_P \leftarrow_{\$} \{0,1\}^{k_P}$
    $C_1 \leftarrow f(K_P)$
    $K \leftarrow \mathsf{H}_2(\langle f \,\|\, R, \boldsymbol{H}[i], K_P\rangle)$
    $\tilde{H} \leftarrow \langle \boldsymbol{H}[i], C_1\rangle$; $AD \leftarrow AD \cup \tilde{H}$
    $C_2 \leftarrow \mathsf{AEAD.Enc}(K, \mathsf{extract}(\tilde{H}), \tilde{H}, \boldsymbol{M}_b[i])$
    $\boldsymbol{C}[i] \leftarrow C_1 \,\|\, C_2$
for $i \leftarrow 1$ to $|\boldsymbol{H}|$ do $Q \leftarrow Q \cup \{(\boldsymbol{H}_i, \boldsymbol{C}_i)\}$
return $\boldsymbol{C}$

$\mathbf{Experiment}\ \mathbf{G}_3(\mathcal{A},k)$:
Is like $\mathbf{G}_2$ except:
$\mathbf{Oracle\ Dec}(H^*, C_1^* \,\|\, C_2^*)$:
if $(H^*, C_1^* \,\|\, C_2^*) \in Q$ then return $\lightning$
if for every $(H, C_1 \,\|\, C_2) \in Q$, $C_1 \neq C_1^*$
    if there is $Y = T_1[pk, H^*, M', X']$
     for some $M', X'$, s.t. $f(Y) = C_1^*$
        then $\tilde{H} \leftarrow \langle H^*, C_1\rangle$, $K \leftarrow \mathsf{H}_2(\langle pk, H^*, Y\rangle)$
        return $M \leftarrow \mathsf{AEAD.Dec}(K, \mathsf{extract}(\tilde{H}), \tilde{H}, C_2^*)$
else return $\mathsf{HE.Dec}_{sk}(H^*, C_1^* \,\|\, C_2^*)$
```

Fig. 14. Games 0-3 in the Proof of Theorem 6: Colored areas indicate the differences between the games.

Experiment $\mathbf{G}_4(\mathcal{A},k)$:
Is like $\mathbf{G}_3$ except:

Oracle Dec$(H^*, C_1^* \| C_2^*)$:

```
if $(H^*, C_1^* \| C_2^*) \in Q$ then return $\lightning$
if for every $(H, C_1 \| C_2) \in Q$, $C_1 \neq C_1^*$
    if there is $Y = T_1[pk, H^*, M', X']$
    for some $M', X'$, s.t. $f(Y) = C_1^*$
        then $\tilde{H} \leftarrow \langle H^*, C_1 \rangle$, $K \leftarrow \mathsf{H}_2(\langle pk, H^*, Y \rangle)$
        return $M \leftarrow \mathsf{AEAD.Dec}(K, \mathsf{extract}(\tilde{H}), \tilde{H}, C_2^*)$
    else $\mathsf{bad}_2 \leftarrow \mathsf{true}$, abort
else return $\mathsf{HE.Dec}_{sk}(H^*, C_1^* \| C_2^*)$
```

Experiment $\mathbf{G}_5(\mathcal{A},k)$:
Is like $\mathbf{G}_4$ except:

Oracle $\mathsf{H}_2(\langle P, H, KP \rangle)$:

```
$Y \leftarrow_\$ \{0,1\}^\lambda$
If $P = pk$ and $T_2[\langle P, H, KP \rangle] \neq \bot$ and $f(KP) = C_1$
for some $(C_1 \| C_2) \in Q$ then $\mathsf{bad}_3 \leftarrow \mathsf{true}$
If $T_2[\langle P, H, KP \rangle] = \bot$ then $T_2[\langle P, H, KP \rangle] \leftarrow Y$
return $T_2[\langle P, H, KP \rangle]$
```

Experiment $\mathbf{G}_6(\mathcal{A},k)$:
Is like $\mathbf{G}_5$ except:

Oracle Dec$(H^*, C_1^* \| C_2^*)$:

```
if $(H^*, C_1^* \| C_2^*) \in Q$ then return $\lightning$
if for every $(H, C_1 \| C_2) \in Q$, $C_1 \neq C_1^*$
    if there is $Y = T_1[pk, H^*, M', X']$
    for some $M', X'$, s.t. $f(Y) = C_1^*$
        then $\tilde{H} \leftarrow \langle H^*, C_1 \rangle$, $K \leftarrow \mathsf{H}_2(\langle pk, H^*, Y \rangle)$
        return $M \leftarrow \mathsf{AEAD.Dec}(K, \mathsf{extract}(\tilde{H}), \tilde{H}, C_2^*)$
    else $\mathsf{bad}_2 \leftarrow \mathsf{true}$, abort
else if $C_1^* = C_1$ for some $(H, C_1 \| C_2) \in Q$
and $\mathsf{HE.Dec}_{sk}(H^*, C_1^* \| C_2^*) \neq \lightning$
then $\mathsf{bad}_4 \leftarrow \mathsf{true}$, abort
return $\mathsf{HE.Dec}_{sk}(H^*, C_1^* \| C_2^*)$
```

Experiment $\mathbf{G}_7(\mathcal{A},k)$:
Is like $\mathbf{G}_6$ except:

Oracle LR$(\boldsymbol{H}, \mathcal{M})$:

```
if $\mathsf{pkout} = \mathsf{true}$ then return $\lightning$
$(\boldsymbol{M}_0, \boldsymbol{M}_1, \boldsymbol{r}) \leftarrow_\$ \mathcal{M}^{\mathsf{H}_1,\mathsf{H}_2}(1^k)$
For $i = 1$ to $v(k)$
    $K_P \leftarrow_\$ \{0,1\}^{k_P}$; $C_1 \leftarrow f(K_P)$; $K \leftarrow_\$ \{0,1\}^\lambda$
    $\tilde{H} \leftarrow \langle \boldsymbol{H}[i], C_1 \rangle$
    if $\tilde{H} \in AD$ then $\mathsf{bad} \leftarrow \mathsf{true}$
    $AD \leftarrow AD \cup \tilde{H}$
    $C_2 \leftarrow \mathsf{AEAD.Enc}(K, \mathsf{extract}(\tilde{H}), \tilde{H}, \boldsymbol{M}_b[i])$
    $\boldsymbol{C}[i] \leftarrow C_1 \| C_2$
for $i \leftarrow 1$ to $|\boldsymbol{H}|$ do $Q \leftarrow Q \cup \{(\boldsymbol{H}_i, \boldsymbol{C}_i)\}$
return $\boldsymbol{C}$
```

Experiment $\mathbf{G}_8(\mathcal{A},k)$:
Is like $\mathbf{G}_7$ except:

Oracle LR$(\boldsymbol{H}, \mathcal{M})$:

```
if $\mathsf{pkout} = \mathsf{true}$ then return $\lightning$
$(\boldsymbol{M}_0, \boldsymbol{M}_1, \boldsymbol{r}) \leftarrow_\$ \mathcal{M}^{\mathsf{H}_1,\mathsf{H}_2}(1^k)$
For $i = 1$ to $v(k)$
    $\boldsymbol{M}_b[i] \leftarrow_\$ \{0,1\}^{n(k)}$
    $K_P \leftarrow \{0,1\}^{k_P}$; $C_1 \leftarrow f(K_P)$
    $K \leftarrow_\$ \{0,1\}^\lambda$
    $\tilde{H} \leftarrow \langle \boldsymbol{H}[i], C_1 \rangle$; $AD \leftarrow AD \cup \tilde{H}$
    $C_2 \leftarrow \mathsf{AEAD.Enc}(K, \mathsf{extract}(\tilde{H}), \tilde{H}, \boldsymbol{M}_b[i])$
    $\boldsymbol{C}[i] \leftarrow C_1 \| C_2$
for $i \leftarrow 1$ to $|\boldsymbol{H}|$ do $Q \leftarrow Q \cup \{(\boldsymbol{H}_i, \boldsymbol{C}_i)\}$
return $\boldsymbol{C}$
```

Experiment $\mathbf{G}_9(\mathcal{A},k)$:
Is like $\mathbf{G}_8$ except:

Oracle Dec$(H^*, C_1^* \| C_2^*)$:

```
if $(H^*, C_1^* \| C_2^*) \in Q$ then return $\lightning$
if for every $(H, C_1 \| C_2) \in Q$, $C_1 \neq C_1^*$
    and $T_2[\langle pk, H^*, K_P \rangle] \neq \bot$ for $f(K_p) = C_1^*$
    and $\mathsf{pkout} = \mathsf{false}$ then $\mathsf{bad}_5 \leftarrow \mathsf{true}$, abort
    else if there is $Y = T_1[pk, H', M', X']$
    for some $H', M', X'$, s.t. $f(Y) = C_1^*$
            then $\tilde{H} \leftarrow \langle H^*, C_1 \rangle$, $K \leftarrow \mathsf{H}_2(\langle pk, H^8, Y \rangle)$
            return $M \leftarrow \mathsf{AEAD.Dec}(K, \mathsf{extract}(\tilde{H}), \tilde{H}, C_2^*)$
        else $\mathsf{bad}_2 \leftarrow \mathsf{true}$, abort
else if $C_1^* = C_1$ for some $(H, C_1 \| C_2) \in Q$
and $\mathsf{HE.Dec}_{sk}(H^*, C_1^* \| C_2^*) \neq \lightning$
then $\mathsf{bad}_4 \leftarrow \mathsf{true}$, abort
return $\mathsf{HE.Dec}_{sk}(H^*, C_1^* \| C_2^*)$
```

Experiment $\mathbf{G}_{10}(\mathcal{A},k)$:
Is like $\mathbf{G}_9$ except:

Oracle $\mathsf{H}_1(\langle P, H, M, X \rangle)$:

```
$Y \leftarrow_\$ \{0,1\}^{k_P}$
If $P = pk$ and $T_1[\langle P, H, M, X \rangle] \neq \bot$
    if $\mathsf{pkout} = \mathsf{true}$ then $\mathsf{bad}_1^a \leftarrow \mathsf{true}$
    if $\mathsf{pkout} = \mathsf{false}$ then $\mathsf{bad}_1^b \leftarrow \mathsf{true}$
If $T_1[\langle P, H, M, X \rangle] = \bot$ then $T_1[\langle P, H, M, X \rangle] \leftarrow Y$
return $T_1[\langle P, H, M, X \rangle]$
```

Fig. 15. Games 4-10 in the Proof of Theorem 6: Colored areas indicate the differences between the games.

```
$\mathcal{B}(1^k, f, y)$
$Q, Q_1, Q_G, Q_2 \leftarrow \emptyset$; $x \leftarrow \bot$
$pk \leftarrow \langle f \rangle$; $j \leftarrow\$ [v]$
$b' \leftarrow\$ \mathcal{A}_1^{\mathbf{LR}', \mathbf{Dec}', \mathsf{H}_1, \mathsf{G}, \mathsf{H}_2}(1^k, pk)$
return $x$

Oracle $\mathbf{LR}'(\boldsymbol{H}, \mathcal{M})$
for $i \leftarrow 1$ to $v$ do
  $Y_\ell \leftarrow\$ \{0,1\}^{n-k_1}$; $Y_r \leftarrow\$ \{0,1\}^{k_1}$
  if $i = j$ then $\boldsymbol{C}[i] \leftarrow Y_\ell \,\|\, y$
  else $\boldsymbol{C}[i] \leftarrow Y_\ell \,\|\, f(Y_r)$
  $Q \leftarrow Q \cup \{(\boldsymbol{H}[i], \boldsymbol{C}[i])\}$
return $\boldsymbol{C}$

Oracle $\mathsf{H}_1(X)$
if $T_1[X] = \bot$ then $T_1[X] \leftarrow\$ \{0,1\}^{k_0}$
$Q_1 \leftarrow Q_1 \cup \{X\}$; return $T_1[X]$
```

```
Oracle $\mathbf{Dec}'(\boldsymbol{H}, \mathcal{M})$
if $(H, C) \in Q$ then return $\lightning$
for each $(A_1, A_G, A_2) \in Q_1 \times Q_G \times Q_2$ do
  $H^* \leftarrow A_G \oplus \mathsf{H}_1(A_1)$
  $Y_\ell \leftarrow C[1..a]$; $Y_r^* \leftarrow A_2[a+1..|A_2|]$
  if $Y_r \,\|\, f(Y_r^*) = C \wedge H = H^*$ then return $A_1$
return $\bot$

Oracle $\mathsf{G}(X)$
if $T_G[X] = \bot$ then $T_G[X] \leftarrow\$ \{0,1\}^{n-k_0}$
$Q_G \leftarrow Q_G \cup \{X\}$; return $T_G[X]$

Oracle $\mathsf{H}_2(X)$
if $T_2[X] = \bot$ then $T_2[X] \leftarrow\$ \{0,1\}^{k_0}$
if $|X| \geq k_1$ then
  $x' \leftarrow X[|X| - k_1 + 1..|X|$  # Last $k_1$ bits
  if $f(x') = y$ then $x \leftarrow x'$
$Q_2 \leftarrow Q_2 \cup \{X\}$; return $T_2[X]$
```

```
$\mathbf{G}_0(\mathcal{A}_1, k)$
$Q \leftarrow \emptyset$; $(pk, sk) \leftarrow\$ \mathsf{Kgen}(1^k)$
$\langle f \rangle \leftarrow pk$ ; $\langle f^{-1} \rangle \leftarrow sk$
$b \leftarrow\$ \{0,1\}$; $b' \leftarrow\$ \mathcal{A}_1^{\mathbf{LR}, \mathbf{Dec}, \mathsf{H}_1, \mathsf{G}, \mathsf{H}_2}(1^k, pk)$
return $b = b'$

Oracle $\mathbf{Dec}(H, C)$:
if $(H, C) \in Q$ then return $\lightning$
if $|Y| < n - k_1$ then return $\bot$
$Y_\ell \leftarrow C[1..a]$; $Y_r \leftarrow f^{-1}(C[a+1..|Y|])$
$S_1 \,\|\, T_0 \leftarrow Y_\ell \,\|\, Y_r$  # where $|S_1| = k_0$
$S_0 \leftarrow \mathsf{H}_2(pk \,\|\, T_0) \oplus S_1$
$X_r \leftarrow \mathsf{G}(pk \,\|\, S_0) \oplus T_0$
$X_\ell \leftarrow \mathsf{H}_1(pk \,\|\, X_r) \oplus S_0$
$H^* \,\|\, M \leftarrow X_\ell \,\|\, X_r$  # where $|H^*| = k_0$
if $H^* \neq H$ then return $\bot$
$\langle M, r \rangle \leftarrow \mathsf{unpad}_{n-k_0}(PM)$; return $M$

Oracle $\mathsf{H}_1(X)$
if $T_1[X] = \bot$ then $T_1[X] \leftarrow\$ \{0,1\}^{k_0}$
for $i \leftarrow 1$ to $v$ do
  if $\mathsf{qr} \wedge X = pk \,\|\, \boldsymbol{X}_r[i]$ then
    $\mathsf{bad}_2 \leftarrow \mathsf{true}$; $T_1[X] \leftarrow \boldsymbol{r}_1[i]$
return $T_1[X]$

Oracle $\mathsf{G}(X)$
if $T_G[X] = \bot$ then $T_G[X] \leftarrow\$ \{0,1\}^{n-k_0}$
for $i \leftarrow 1$ to $v$ do
  if $\mathsf{qr} \wedge X = pk \,\|\, \boldsymbol{S}_0[i]$ then
    $\mathsf{bad}_3 \leftarrow \mathsf{true}$; $T_G[X] \leftarrow \boldsymbol{r}_G[i]$
return $T_G[X]$
```

```
Oracle $\mathbf{LR}(\boldsymbol{H}, \mathcal{M})$:                                  $\mathbf{G}_1(\mathcal{A}_1, k)$
$\boldsymbol{r} \leftarrow\$ \mathcal{R}^{\mathsf{H}_1, \mathsf{G}, \mathsf{H}_2}(1^k)$
$(\boldsymbol{M}_0, \boldsymbol{M}_1) \leftarrow\$ \mathcal{M}^{\mathsf{H}_1, \mathsf{G}, \mathsf{H}_2}(1^k)$
for $i \leftarrow 1$ to $v$ do
  $X \leftarrow \boldsymbol{H}[i] \,\|\, \mathsf{pad}_{n-k_0}(\langle \boldsymbol{M}_b[i], \boldsymbol{r}[i] \rangle)$
  $\boldsymbol{X}_\ell[i] \leftarrow X[1..k_0]$; $\boldsymbol{X}_r[i] \leftarrow X[k_0 + 1..|X|]$
  $\boldsymbol{r}_1[i], \boldsymbol{r}_2[i] \leftarrow\$ \{0,1\}^{k_0}$
  $\boldsymbol{r}_G[i] \leftarrow\$ \{0,1\}^{n-k_0}$
  if $T_1[pk \,\|\, \boldsymbol{X}_r[i]]$ defined then
    $\mathsf{bad}_{1,1} \leftarrow \mathsf{true}$; $\boldsymbol{r}_1[i] \leftarrow T_1[pk \,\|\, \boldsymbol{X}_r[i]]$
  $\boldsymbol{S}_0[i] \leftarrow$ $\mathsf{H}_1(pk \,\|\, \boldsymbol{X}_r[i])$ $\boldsymbol{r}_1[i]$ $\oplus \boldsymbol{X}_\ell[i]$
  if $T_G[pk \,\|\, \boldsymbol{S}_0[i]]$ defined then
    $\mathsf{bad}_{1,2} \leftarrow \mathsf{true}$; $\boldsymbol{r}_G[i] \leftarrow T_G[pk \,\|\, \boldsymbol{S}_0[i]]$
  $\boldsymbol{T}_0[i] \leftarrow$ $\mathsf{G}(pk \,\|\, \boldsymbol{S}_0[i])$ $\boldsymbol{r}_G[i]$ $\oplus \boldsymbol{X}_r[i]$
  if $T_G[pk \,\|\, \boldsymbol{T}_0[i]]$ defined then
    $\mathsf{bad}_{1,3} \leftarrow \mathsf{true}$; $\boldsymbol{r}_2[i] \leftarrow T_2[pk \,\|\, \boldsymbol{T}_0[i]]$
  $\boldsymbol{S}_1[i] \leftarrow$ $\mathsf{H}_2(pk \,\|\, \boldsymbol{T}_0[i])$ $\boldsymbol{r}_2[i]$ $\oplus \boldsymbol{S}_0[i]$
  $Y_\ell \,\|\, Y_r \leftarrow \boldsymbol{S}_1[i] \,\|\, \boldsymbol{T}_0[i]$  # where $|Y_r| = k_1$
  $\boldsymbol{C}[i] \leftarrow Y_\ell \,\|\, f(Y_r)$
for $i \leftarrow 1$ to $v$ do $Q \leftarrow Q \cup \{(\boldsymbol{H}_i, \boldsymbol{C}_i)\}$
$\mathsf{qr} \leftarrow \mathsf{true}$; return $\boldsymbol{C}$

Oracle $\mathsf{H}_2(X)$
if $T_2[X] = \bot$ then $T_2[X] \leftarrow\$ \{0,1\}^{k_0}$
for $i \leftarrow 1$ to $v$ do
  if $\mathsf{qr} \wedge X = pk \,\|\, \boldsymbol{T}_0[i]$ then
    $\mathsf{bad}^* \leftarrow \mathsf{true}$; $T_2[X] \leftarrow \boldsymbol{r}_2[i]$
return $T_2[X]$
```

Fig. 16. Games 0 and 1 and adversary $\mathcal{B}$ for proof of Theorem 7. Let $a = \max\{0, n - k_1\}$.

```
Oracle $\mathbf{LR}(\boldsymbol{H}, \mathcal{M})$:                                  $\mathbf{G}_2(\mathcal{A}_1, k)$
$\boldsymbol{r} \leftarrow_{\$} \mathcal{R}^{\mathsf{H}_1,\mathsf{G},\mathsf{H}_2}(1^k)$
$(\boldsymbol{M}_0, \boldsymbol{M}_1) \leftarrow_{\$} \mathcal{M}^{\mathsf{H}_1,\mathsf{G},\mathsf{H}_2}(1^k)$
for $i \leftarrow 1$ to $v$ do
  $X \leftarrow \boldsymbol{H}[i] \,\|\, \mathsf{pad}_{n-k_0}(\langle \boldsymbol{M}_b[i], \boldsymbol{r}[i] \rangle)$
  $\boldsymbol{X}_\ell[i] \leftarrow X[1..k_0]$
  $\boldsymbol{X}_r[i] \leftarrow X[k_0+1..|X|]$
  $\boldsymbol{r}_1[i], \boldsymbol{r}_2[i] \leftarrow_{\$} \{0,1\}^{k_0}$
  $\boldsymbol{r}_G[i] \leftarrow_{\$} \{0,1\}^{n-k_0}$
  if $T_1[pk \,\|\, \boldsymbol{X}_r[i]]$ defined then
    $\mathsf{bad}_{1,1} \leftarrow \mathsf{true}$; $\boldsymbol{r}_1[i] \leftarrow T_1[pk \,\|\, \boldsymbol{X}_r[i]]$
  $\boldsymbol{S}_0[i] \leftarrow \boldsymbol{r}_1[i] \oplus \boldsymbol{X}_\ell[i]$
  if $T_G[pk \,\|\, \boldsymbol{S}_0[i]]$ defined then
    $\mathsf{bad}_{1,2} \leftarrow \mathsf{true}$; $\boldsymbol{r}_G[i] \leftarrow T_G[pk \,\|\, \boldsymbol{S}_0[i]]$
  $\boldsymbol{T}_0[i] \leftarrow \boldsymbol{r}_G[i] \oplus \boldsymbol{X}_r[i]$
  if $T_G[pk \,\|\, \boldsymbol{T}_0[i]]$ defined then
    $\mathsf{bad}_{1,3} \leftarrow \mathsf{true}$; $\boldsymbol{r}_2[i] \leftarrow T_2[pk \,\|\, \boldsymbol{T}_0[i]]$
  $\boldsymbol{S}_1[i] \leftarrow \boldsymbol{r}_2[i] \oplus \boldsymbol{S}_0[i]$
  $Y_\ell \,\|\, Y_r \leftarrow \boldsymbol{S}_1[i] \,\|\, \boldsymbol{T}_0[i]$  # where $|Y_r| = k_1$
  $\boldsymbol{C}[i] \leftarrow Y_\ell \,\|\, f(Y_r)$
for $i \leftarrow 1$ to $v$ do $Q \leftarrow Q \cup \{(\boldsymbol{H}_i, \boldsymbol{C}_i)\}$
$\mathsf{qr} \leftarrow \mathsf{true}$; return $\boldsymbol{C}$
```

```
Oracle $\mathbf{LR}(\boldsymbol{H}, \mathcal{M})$:                                  $\mathbf{G}_3(\mathcal{A}_1, k)$
$\boldsymbol{r} \leftarrow_{\$} \mathcal{R}^{\mathsf{H}_1,\mathsf{G},\mathsf{H}_2}(1^k)$
$(\boldsymbol{M}_0, \boldsymbol{M}_1) \leftarrow_{\$} \mathcal{M}^{\mathsf{H}_1,\mathsf{G},\mathsf{H}_2}(1^k)$
for $i \leftarrow 1$ to $v$ do
  $X \leftarrow \boldsymbol{H}[i] \,\|\, \mathsf{pad}_{n-k_0}(\langle \boldsymbol{M}_b[i], \boldsymbol{r}[i] \rangle)$
  $\boldsymbol{X}_\ell[i] \leftarrow X[1..k_0]$
  $\boldsymbol{X}_r[i] \leftarrow X[k_0+1..|X|]$
  $\boldsymbol{r}_1[i], \boldsymbol{r}_2[i] \leftarrow_{\$} \{0,1\}^{k_0}$
  $\boldsymbol{r}_G[i] \leftarrow_{\$} \{0,1\}^{n-k_0}$
  if $T_1[pk \,\|\, \boldsymbol{X}_r[i]]$ defined then $\mathsf{bad}_{1,1} \leftarrow \mathsf{true}$
  $\boldsymbol{S}_0[i] \leftarrow \boldsymbol{r}_1[i] \oplus \boldsymbol{X}_\ell[i]$
  if $T_G[pk \,\|\, \boldsymbol{S}_0[i]]$ defined then $\mathsf{bad}_{1,2} \leftarrow \mathsf{true}$
  $\boldsymbol{T}_0[i] \leftarrow \boldsymbol{r}_G[i] \oplus \boldsymbol{X}_r[i]$
  if $T_G[pk \,\|\, \boldsymbol{T}_0[i]]$ defined then $\mathsf{bad}_{1,3} \leftarrow \mathsf{true}$
  $\boldsymbol{S}_1[i] \leftarrow \boldsymbol{r}_2[i] \oplus \boldsymbol{S}_0[i]$
  $Y_\ell \,\|\, Y_r \leftarrow \boldsymbol{S}_1[i] \,\|\, \boldsymbol{T}_0[i]$  # where $|Y_r| = k_1$
  $\boldsymbol{C}[i] \leftarrow Y_\ell \,\|\, f(Y_r)$
for $i \leftarrow 1$ to $v$ do $Q \leftarrow Q \cup \{(\boldsymbol{H}_i, \boldsymbol{C}_i)\}$
$\mathsf{qr} \leftarrow \mathsf{true}$; return $\boldsymbol{C}$

Oracle $\mathsf{H}_1(X)$
if $T_1[X] = \bot$ then $T_1[X] \leftarrow_{\$} \{0,1\}^{k_0}$
for $i \leftarrow 1$ to $v$ do
  if $\mathsf{qr} \wedge X = pk \,\|\, \boldsymbol{X}_r[i]$ then
    $\mathsf{bad}_2 \leftarrow \mathsf{true}$; $T_1[X] \leftarrow \boldsymbol{r}_1[i]$
return $T_1[X]$
```

```
Oracle $\mathbf{LR}(\boldsymbol{H}, \mathcal{M})$:                                  $\mathbf{G}_4(\mathcal{A}_1, k)$
$\boldsymbol{r} \leftarrow_{\$} \mathcal{R}^{\mathsf{H}_1,\mathsf{G},\mathsf{H}_2}(1^k)$
$(\boldsymbol{M}_0, \boldsymbol{M}_1) \leftarrow_{\$} \mathcal{M}^{\mathsf{H}_1,\mathsf{G},\mathsf{H}_2}(1^k)$
for $i \leftarrow 1$ to $v$ do
  $X \leftarrow \boldsymbol{H}[i] \,\|\, \mathsf{pad}_{n-k_0}(\langle \boldsymbol{M}_b[i], \boldsymbol{r}[i] \rangle)$
  $\boldsymbol{X}_r[i] \leftarrow X[k_0+1..|X|]$
  $\boldsymbol{r}_1[i], \boldsymbol{r}_2[i] \leftarrow_{\$} \{0,1\}^{k_0}$
  $\boldsymbol{r}_G[i] \leftarrow_{\$} \{0,1\}^{n-k_0}$
  $\boldsymbol{S}_0[i] \leftarrow \boldsymbol{r}_1[i]$
  $\boldsymbol{T}_0[i] \leftarrow \boldsymbol{r}_G[i] \oplus \boldsymbol{X}_r[i]$
  $\boldsymbol{S}_1[i] \leftarrow \boldsymbol{r}_2[i] \oplus \boldsymbol{S}_0[i]$
  $Y_\ell \,\|\, Y_r \leftarrow \boldsymbol{S}_1[i] \,\|\, \boldsymbol{T}_0[i]$  # where $|Y_r| = k_1$
  $\boldsymbol{C}[i] \leftarrow Y_\ell \,\|\, f(Y_r)$
for $i \leftarrow 1$ to $v$ do $Q \leftarrow Q \cup \{(\boldsymbol{H}_i, \boldsymbol{C}_i)\}$
$\mathsf{qr} \leftarrow \mathsf{true}$; return $\boldsymbol{C}$

Oracle $\mathsf{G}(X)$
if $T_G[X] = \bot$ then $T_G[X] \leftarrow_{\$} \{0,1\}^{n-k_0}$
for $i \leftarrow 1$ to $v$ do
  if $\mathsf{qr} \wedge X = pk \,\|\, \boldsymbol{S}_0[i]$ then
    $\mathsf{bad}_3 \leftarrow \mathsf{true}$; $T_G[X] \leftarrow \boldsymbol{r}_G[i]$
return $T_G[X]$
```

```
Oracle $\mathbf{LR}(\boldsymbol{H}, \mathcal{M})$:                                  $\mathbf{G}_5(\mathcal{A}_1, k)$
$\boldsymbol{r} \leftarrow_{\$} \mathcal{R}^{\mathsf{H}_1,\mathsf{G},\mathsf{H}_2}(1^k)$
$(\boldsymbol{M}_0, \boldsymbol{M}_1) \leftarrow_{\$} \mathcal{M}^{\mathsf{H}_1,\mathsf{G},\mathsf{H}_2}(1^k)$
for $i \leftarrow 1$ to $v$ do
  $\boldsymbol{r}_1[i], \boldsymbol{r}_2[i] \leftarrow_{\$} \{0,1\}^{k_0}$
  $\boldsymbol{r}_G[i] \leftarrow_{\$} \{0,1\}^{n-k_0}$
  $\boldsymbol{S}_0[i] \leftarrow \boldsymbol{r}_1[i]$
  $\boldsymbol{T}_0[i] \leftarrow \boldsymbol{r}_G[i]$
  $\boldsymbol{S}_1[i] \leftarrow \boldsymbol{r}_2[i] \oplus \boldsymbol{S}_0[i]$
  $Y_\ell \,\|\, Y_r \leftarrow \boldsymbol{S}_1[i] \,\|\, \boldsymbol{T}_0[i]$  # where $|Y_r| = k_1$
  $\boldsymbol{C}[i] \leftarrow Y_\ell \,\|\, f(Y_r)$
for $i \leftarrow 1$ to $v$ do $Q \leftarrow Q \cup \{(\boldsymbol{H}_i, \boldsymbol{C}_i)\}$
$\mathsf{qr} \leftarrow \mathsf{true}$; return $\boldsymbol{C}$

Oracle $\mathsf{H}_2(X)$
if $T_2[X] = \bot$ then $T_2[X] \leftarrow_{\$} \{0,1\}^{k_0}$
for $i \leftarrow 1$ to $v$ do
  if $\mathsf{qr} \wedge X = pk \,\|\, \boldsymbol{T}_0[i]$ then
    $\mathsf{bad}^* \leftarrow \mathsf{true}$; $T_2[X] \leftarrow \boldsymbol{r}_2[i]$
return $T_2[X]$
```

Fig. 17. Games 2-5 for proof of Theorem 7.

```
**Oracle Dec**$(H, C)$:                                    $\mathbf{G}_6(\mathcal{A}_1, k)$
if $(H, C) \in Q$ then return $\lightning$
if $|Y| < n - k_1$ then return $\bot$
$Y_\ell \leftarrow C[1..a]$; $Y_r \leftarrow f^{-1}(C[a+1..|Y|])$
$S_1 \| T_0 \leftarrow Y_\ell \| Y_r$  # where $|S_1| = k_0$
if $T_2[pk \| T_0]$ is undefined then $\mathsf{bad}_6' \leftarrow \mathsf{true}$
$S_0 \leftarrow \mathsf{H}_2(pk \| T_0) \oplus S_1$
if $T_G[pk \| S_0]$ is undefined then $\mathsf{bad}_8' \leftarrow \mathsf{true}$
$X_r \leftarrow \mathsf{G}(pk \| S_0) \oplus T_0$
if $T_1[pk \| X_r]$ is undefined then $\mathsf{bad}_7' \leftarrow \mathsf{true}$
$X_\ell \leftarrow \mathsf{H}_1(pk \| X_r) \oplus S_0$
$H^* \| M \leftarrow X_\ell \| X_r$  # where $|H^*| = k_0$
if $H^* \neq H$ then return $\bot$
else if $\mathsf{bad}_6'$ then $\mathsf{bad}_6 \leftarrow \mathsf{true}$
else if $\mathsf{bad}_7'$ then $\mathsf{bad}_7 \leftarrow \mathsf{true}$
else if $\mathsf{bad}_8'$ then $\mathsf{bad}_8 \leftarrow \mathsf{true}$
$\langle M, r \rangle \leftarrow \mathsf{unpad}_{n-k_0}(PM)$; return $M$
```

```
**Oracle Dec**$(H, C)$:                                    $\mathbf{G}_7(\mathcal{A}_1, k)$
if $(H, C) \in Q$ then return $\lightning$
if $|Y| < n - k_1$ then return $\bot$
$Y_\ell \leftarrow C[1..a]$; $Y_r \leftarrow f^{-1}(C[a+1..|Y|])$
$S_1 \| T_0 \leftarrow Y_\ell \| Y_r$  # where $|S_1| = k_0$
if $T_2[pk \| T_0]$ is undefined then $\mathsf{bad}_6' \leftarrow \mathsf{true}$
$S_0 \leftarrow \mathsf{H}_2(pk \| T_0) \oplus S_1$
if $T_G[pk \| S_0]$ is undefined then $\mathsf{bad}_8' \leftarrow \mathsf{true}$
$X_r \leftarrow \mathsf{G}(pk \| S_0) \oplus T_0$
if $T_1[pk \| X_r]$ is undefined then $\mathsf{bad}_7' \leftarrow \mathsf{true}$
$X_\ell \leftarrow \mathsf{H}_1(pk \| X_r) \oplus S_0$
$H^* \| M \leftarrow X_\ell \| X_r$  # where $|H^*| = k_0$
if $H^* \neq H$ then return $\bot$
else if $\mathsf{bad}_6'$ then $\mathsf{bad}_6 \leftarrow \mathsf{true}$; return $\bot$
else if $\mathsf{bad}_7'$ then $\mathsf{bad}_7 \leftarrow \mathsf{true}$
else if $\mathsf{bad}_8'$ then $\mathsf{bad}_8 \leftarrow \mathsf{true}$
$\langle M, r \rangle \leftarrow \mathsf{unpad}_{n-k_0}(PM)$; return $M$
```

```
**Oracle Dec**$(H, C)$:                                    $\mathbf{G}_8(\mathcal{A}_1, k)$
if $(H, C) \in Q$ then return $\lightning$
if $|Y| < n - k_1$ then return $\bot$
$Y_\ell \leftarrow C[1..a]$; $Y_r \leftarrow f^{-1}(C[a+1..|Y|])$
$S_1 \| T_0 \leftarrow Y_\ell \| Y_r$  # where $|S_1| = k_0$
if $T_2[pk \| T_0]$ is undefined then $\mathsf{bad}_6' \leftarrow \mathsf{true}$
$S_0 \leftarrow \mathsf{H}_2(pk \| T_0) \oplus S_1$
if $T_G[pk \| S_0]$ is undefined then $\mathsf{bad}_8' \leftarrow \mathsf{true}$
$X_r \leftarrow \mathsf{G}(pk \| S_0) \oplus T_0$
if $T_1[pk \| X_r]$ is undefined then $\mathsf{bad}_7' \leftarrow \mathsf{true}$
$X_\ell \leftarrow \mathsf{H}_1(pk \| X_r) \oplus S_0$
$H^* \| M \leftarrow X_\ell \| X_r$  # where $|H^*| = k_0$
if $H^* \neq H$ then return $\bot$
else if $\mathsf{bad}_6'$ then $\mathsf{bad}_6 \leftarrow \mathsf{true}$; return $\bot$
else if $\mathsf{bad}_7'$ then $\mathsf{bad}_7 \leftarrow \mathsf{true}$; return $\bot$
else if $\mathsf{bad}_8'$ then $\mathsf{bad}_8 \leftarrow \mathsf{true}$
$\langle M, r \rangle \leftarrow \mathsf{unpad}_{n-k_0}(PM)$; return $M$
```

```
**Oracle Dec**$(H, C)$:                                    $\mathbf{G}_9(\mathcal{A}_1, k)$
if $(H, C) \in Q$ then return $\lightning$
if $|Y| < n - k_1$ then return $\bot$
$Y_\ell \leftarrow C[1..a]$; $Y_r \leftarrow f^{-1}(C[a+1..|Y|])$
$S_1 \| T_0 \leftarrow Y_\ell \| Y_r$  # where $|S_1| = k_0$
if $T_2[pk \| T_0]$ is undefined then $\mathsf{bad}_6' \leftarrow \mathsf{true}$
$S_0 \leftarrow \mathsf{H}_2(pk \| T_0) \oplus S_1$
if $T_G[pk \| S_0]$ is undefined then $\mathsf{bad}_8' \leftarrow \mathsf{true}$
$X_r \leftarrow \mathsf{G}(pk \| S_0) \oplus T_0$
if $T_1[pk \| X_r]$ is undefined then $\mathsf{bad}_7' \leftarrow \mathsf{true}$
$X_\ell \leftarrow \mathsf{H}_1(pk \| X_r) \oplus S_0$
$H^* \| M \leftarrow X_\ell \| X_r$  # where $|H^*| = k_0$
if $H^* \neq H$ then return $\bot$
else if $\mathsf{bad}_6'$ then $\mathsf{bad}_6 \leftarrow \mathsf{true}$; return $\bot$
else if $\mathsf{bad}_7'$ then $\mathsf{bad}_7 \leftarrow \mathsf{true}$; return $\bot$
else if $\mathsf{bad}_8'$ then $\mathsf{bad}_8 \leftarrow \mathsf{true}$; return $\bot$
$\langle M, r \rangle \leftarrow \mathsf{unpad}_{n-k_0}(PM)$; return $M$
```

Fig. 18. Games 6-9 for proof of Theorem 7. Let $a = \max\{0, n - k_1\}$.

```
$\mathbf{G}_0(\mathcal{A}_1, k)$
$(f, f^{-1}) \leftarrow_{\$} F(1^k);\ pk_d \leftarrow \langle f \rangle$
$(pk_r, sk_r) \leftarrow_{\$} \mathcal{K}_r(1^k)$
$b \leftarrow_{\$} \{0,1\}$
$b' \leftarrow_{\$} \mathcal{A}_1^{\mathbf{LR},\mathsf{H}_1,\mathsf{G},\mathsf{H}_2}(1^k, \langle pk_r, pk_d \rangle)$
return $b = b'$

Oracle $\mathsf{H}_1(X)$
if $T_1[X] = \bot$ then $T_1[X] \leftarrow_{\$} \{0,1\}^{k_0}$
if $\mathsf{qr} \wedge (\exists\, i \in [v])\ X = pk_d \,\|\, \boldsymbol{X}_r[i]$ then
    $T_1[X] \leftarrow \boldsymbol{r}_1[i]$
return $T_1[X]$

Oracle $\mathsf{G}(X)$
if $T_G[X] = \bot$ then $T_G[X] \leftarrow_{\$} \{0,1\}^{n-k_0}$
if $\mathsf{qr} \wedge (\exists\, i \in [v])\ X = pk_d \,\|\, \boldsymbol{S}_0[i]$ then
    $T_G[X] \leftarrow \boldsymbol{r}_G[i]$
return $T_G[X]$

Oracle $\mathsf{H}_2(X)$
if $T_2[X] = \bot$ then $T_2[X] \leftarrow_{\$} \{0,1\}^{k_0}$
if $\mathsf{qr} \wedge (\exists\, i \in [v])\ X = pk_d \,\|\, \boldsymbol{T}_0[i]$ then
    $T_2[X] \leftarrow \boldsymbol{r}_2[i]$
return $T_2[X]$
```

```
Oracle $\mathbf{LR}(\boldsymbol{H}, \mathcal{M})$                                   $\mathbf{G}_1(\mathcal{A}_1, k)$
$(\boldsymbol{M}_0, \boldsymbol{M}_1) \leftarrow_{\$} \mathcal{M}^{\mathsf{H}_1,\mathsf{G},\mathsf{H}_2}(1^k)$
$\boldsymbol{r} \leftarrow_{\$} \mathcal{R}^{\mathsf{H}_1,\mathsf{G},\mathsf{H}_2}(1^k)$
for $i \leftarrow 1$ to $v$ do
    $X \leftarrow \mathsf{pad}_n(\mathcal{E}_r(pk_r, \boldsymbol{H}[i], \boldsymbol{M}_b[i]\,;\boldsymbol{r}[i]))$
    $\boldsymbol{X}_\ell[i] \leftarrow X[1..|k_0|]$
    $\boldsymbol{X}_r[i] \leftarrow X[|k_0|+1..|X|]$
    $\boldsymbol{r}_1[i], \boldsymbol{r}_2[i] \leftarrow_{\$} \{0,1\}^{k_0}$
    $\boldsymbol{r}_G[i] \leftarrow_{\$} \{0,1\}^{n-k_0}$
    $a_1 \leftarrow T_1[pk_d \,\|\, \boldsymbol{X}_r[i]]$
    if $a_1 \neq \bot$ then $a_G \leftarrow T_G[pk_d \,\|\, (a_1 \oplus \boldsymbol{X}_\ell[i])]$
    else $a_G \leftarrow T_G[pk_d \,\|\, (\boldsymbol{r}_1[i] \oplus \boldsymbol{X}_\ell[i])]$
    if $a_G \neq \bot$ then $a_2 \leftarrow T_2[pk_d \,\|\, (a_G \oplus \boldsymbol{X}_r[i])]$
    else $a_2 \leftarrow T_2[pk_d \,\|\, (\boldsymbol{r}_G[i] \oplus \boldsymbol{X}_r[i])]$
    if $a_1 \neq \bot$ then
    if $a_G \neq \bot$ then $\mathsf{bad}_1 \leftarrow \mathsf{true};\ \boldsymbol{r}_1[i] \leftarrow a_1$
    else $\mathsf{bad}_{3,2} \leftarrow \mathsf{true};\ \boldsymbol{r}_1[i] \leftarrow a_1$
    $\boldsymbol{S}_0[i] \leftarrow \mathsf{H}_1[pk_d \,\|\, \boldsymbol{X}_r[i]]$  $\boldsymbol{r}_1[i]$  $\oplus \boldsymbol{X}_\ell[i]$
    if $a_G \neq \bot$ then
        if $a_1 \neq \bot$ then $\mathsf{bad}_1 \leftarrow \mathsf{true};\ \boldsymbol{r}_G[i] \leftarrow a_G$
        else $\mathsf{bad}_2 \leftarrow \mathsf{true};\ \boldsymbol{r}_G[i] \leftarrow a_G$
    $\boldsymbol{T}_0[i] \leftarrow \mathsf{G}(pk_d \,\|\, \boldsymbol{S}_0[i])$  $\boldsymbol{r}_G[i]$  $\oplus \boldsymbol{X}_r[i]$
    if $a_2 \neq \bot$ then
        if $a_1 \neq \bot \wedge a_G \neq \bot$ then
            $\mathsf{bad}_1 \leftarrow \mathsf{true};\ \boldsymbol{r}_2[i] \leftarrow a_2$
        else if $a_1 = \bot \wedge a_G \neq \bot$ then
            $\mathsf{bad}_2 \leftarrow \mathsf{true};\ \boldsymbol{r}_2[i] \leftarrow a_2$
        else if $a_1 \neq \bot \wedge a_G = \bot$ then
            $\mathsf{bad}_{3,1} \leftarrow \mathsf{true};\ \boldsymbol{r}_2[i] \leftarrow a_2$
        else $\mathsf{bad}_4 \leftarrow \mathsf{true};\ \boldsymbol{r}_2[i] \leftarrow a_2$
    $\boldsymbol{S}_1[i] \leftarrow \mathsf{H}_2(pk \,\|\, \boldsymbol{T}_0[i])$  $\boldsymbol{r}_2[i]$  $\oplus \boldsymbol{S}_0[i]$
    $Y_\ell \leftarrow (\boldsymbol{S}_1[i] \,\|\, \boldsymbol{T}_0[i])[1..|X| - k_1]$
    $Y_r \leftarrow (\boldsymbol{S}_1[i] \,\|\, \boldsymbol{T}_0[i])[|X| - k_1 + 1..|X|]$
    $\boldsymbol{C}[i] \leftarrow Y_\ell \,\|\, f(Y_r)$
$\mathsf{qr} \leftarrow \mathsf{true}$
return $\boldsymbol{C}$
```

Fig. 19. Games 0 and 1 for proof of Theorem 9.

Oracle LR($\boldsymbol{H}, \mathcal{M}$) $\mathbf{G}_2(\mathcal{A}_1, k)$

```
$(\boldsymbol{M}_0, \boldsymbol{M}_1) \leftarrow\$ \mathcal{M}^{\mathsf{H}_1,\mathsf{G},\mathsf{H}_2}(1^k)$
$\boldsymbol{r} \leftarrow\$ \mathcal{R}^{\mathsf{H}_1,\mathsf{G},\mathsf{H}_2}(1^k)$
for $i \leftarrow 1$ to $v$ do
  $X \leftarrow \mathsf{pad}_n(\mathcal{E}_r(pk_r, \boldsymbol{H}[i], \boldsymbol{M}_b[i]\,;\boldsymbol{r}[i]))$
  $\boldsymbol{X}_\ell[i] \leftarrow X[1..|k_0|]$
  $\boldsymbol{X}_r[i] \leftarrow X[|k_0|+1..|X|]$
  $\boldsymbol{r}_1[i], \boldsymbol{r}_2[i] \leftarrow\$ \{0,1\}^{k_0}$
  $\boldsymbol{r}_G[i] \leftarrow\$ \{0,1\}^{n-k_0}$
  $a_1 \leftarrow T_1[pk_d \,\|\, \boldsymbol{X}_r[i]]$
  if $a_1 \neq \perp$ then $a_G \leftarrow T_G[pk_d \,\|\, (a_1 \oplus \boldsymbol{X}_\ell[i])]$
  else $a_G \leftarrow T_G[pk_d \,\|\, (\boldsymbol{r}_1[i] \oplus \boldsymbol{X}_\ell[i])]$
  if $a_G \neq \perp$ then $a_2 \leftarrow T_2[pk_d \,\|\, (a_G \oplus \boldsymbol{X}_r[i])]$
  else $a_2 \leftarrow T_2[pk_d \,\|\, (\boldsymbol{r}_G[i] \oplus \boldsymbol{X}_r[i])]$
  if $a_1 \neq \perp$ then
    if $a_G \neq \perp$ then $\mathsf{bad}_1 \leftarrow \mathsf{true}$; $\boldsymbol{r}_1[i] \leftarrow a_1$
    else $\mathsf{bad}_{3,2} \leftarrow \mathsf{true}$; $\boldsymbol{r}_1[i] \leftarrow a_1$
  $\boldsymbol{S}_0[i] \leftarrow \boldsymbol{r}_1[i] \oplus \boldsymbol{X}_\ell[i]$
  if $a_G \neq \perp$ then
    if $a_1 \neq \perp$ then $\mathsf{bad}_1 \leftarrow \mathsf{true}$; $\boldsymbol{r}_G[i] \leftarrow a_G$
    else $\mathsf{bad}_2 \leftarrow \mathsf{true}$; $\boldsymbol{r}_G[i] \leftarrow a_G$
  $\boldsymbol{T}_0[i] \leftarrow \boldsymbol{r}_G[i] \oplus \boldsymbol{X}_r[i]$
  if $a_2 \neq \perp$ then
    if $a_1 \neq \perp \wedge a_G \neq \perp$ then
      $\mathsf{bad}_1 \leftarrow \mathsf{true}$; $\boldsymbol{r}_2[i] \leftarrow a_2$
    else if $a_1 = \perp \wedge a_G \neq \perp$ then
      $\mathsf{bad}_2 \leftarrow \mathsf{true}$; $\boldsymbol{r}_2[i] \leftarrow a_2$
    else if $a_1 \neq \perp \wedge a_G = \perp$ then
      $\mathsf{bad}_{3,1} \leftarrow \mathsf{true}$; $\boldsymbol{r}_2[i] \leftarrow a_2$
    else $\mathsf{bad}_4 \leftarrow \mathsf{true}$; $\boldsymbol{r}_2[i] \leftarrow a_2$
  $\boldsymbol{S}_1[i] \leftarrow \boldsymbol{r}_2[i] \oplus \boldsymbol{S}_0[i]$
  $Y_\ell \leftarrow (\boldsymbol{S}_1[i] \,\|\, \boldsymbol{T}_0[i])[1..|X|-k_1]$
  $Y_r \leftarrow (\boldsymbol{S}_1[i] \,\|\, \boldsymbol{T}_0[i])[|X|-k_1+1..|X|]$
  $\boldsymbol{C}[i] \leftarrow Y_\ell \,\|\, f(Y_r)$
$\mathsf{qr} \leftarrow \mathsf{true}$
return $\boldsymbol{C}$
```

Oracle LR($\boldsymbol{H}, \mathcal{M}$) $\mathbf{G}_3(\mathcal{A}_1, k)$

```
$(\boldsymbol{M}_0, \boldsymbol{M}_1) \leftarrow\$ \mathcal{M}^{\mathsf{H}_1,\mathsf{G},\mathsf{H}_2}(1^k)$
$\boldsymbol{r} \leftarrow\$ \mathcal{R}^{\mathsf{H}_1,\mathsf{G},\mathsf{H}_2}(1^k)$
for $i \leftarrow 1$ to $v$ do
  $X \leftarrow \mathsf{pad}_n(\mathcal{E}_r(pk_r, \boldsymbol{H}[i], \boldsymbol{M}_b[i]\,;\boldsymbol{r}[i]))$
  $\boldsymbol{X}_\ell[i] \leftarrow X[1..|k_0|]$
  $\boldsymbol{X}_r[i] \leftarrow X[|k_0|+1..|X|]$
  $\boldsymbol{r}_1[i], \boldsymbol{r}_2[i] \leftarrow\$ \{0,1\}^{k_0}$
  $\boldsymbol{r}_G[i] \leftarrow\$ \{0,1\}^{n-k_0}$
  $a_1 \leftarrow T_1[pk_d \,\|\, \boldsymbol{X}_r[i]]$
  if $a_1 \neq \perp$ then $a_G \leftarrow T_G[pk_d \,\|\, (a_1 \oplus \boldsymbol{X}_\ell[i])]$
  else $a_G \leftarrow T_G[pk_d \,\|\, (\boldsymbol{r}_1[i] \oplus \boldsymbol{X}_\ell[i])]$
  if $a_G \neq \perp$ then $a_2 \leftarrow T_2[pk_d \,\|\, (a_G \oplus \boldsymbol{X}_r[i])]$
  else $a_2 \leftarrow T_2[pk_d \,\|\, (\boldsymbol{r}_G[i] \oplus \boldsymbol{X}_r[i])]$
  if $a_1 \neq \perp$ then
    if $a_G \neq \perp$ then $\mathsf{bad}_1 \leftarrow \mathsf{true}$
    else $\mathsf{bad}_{3,2} \leftarrow \mathsf{true}$; $\boldsymbol{r}_1[i] \leftarrow a_1$
  $\boldsymbol{S}_0[i] \leftarrow \boldsymbol{r}_1[i] \oplus \boldsymbol{X}_\ell[i]$
  if $a_G \neq \perp$ then
    if $a_1 \neq \perp$ then $\mathsf{bad}_1 \leftarrow \mathsf{true}$
    else $\mathsf{bad}_2 \leftarrow \mathsf{true}$; $\boldsymbol{r}_G[i] \leftarrow a_G$
  $\boldsymbol{T}_0[i] \leftarrow \boldsymbol{r}_G[i] \oplus \boldsymbol{X}_r[i]$
  if $a_2 \neq \perp$ then
    if $a_1 \neq \perp \wedge a_G \neq \perp$ then
      $\mathsf{bad}_1 \leftarrow \mathsf{true}$
    else if $a_1 = \perp \wedge a_G \neq \perp$ then
      $\mathsf{bad}_2 \leftarrow \mathsf{true}$; $\boldsymbol{r}_2[i] \leftarrow a_2$
    else if $a_1 \neq \perp \wedge a_G = \perp$ then
      $\mathsf{bad}_{3,1} \leftarrow \mathsf{true}$; $\boldsymbol{r}_2[i] \leftarrow a_2$
    else $\mathsf{bad}_4 \leftarrow \mathsf{true}$; $\boldsymbol{r}_2[i] \leftarrow a_2$
  $\boldsymbol{S}_1[i] \leftarrow \boldsymbol{r}_2[i] \oplus \boldsymbol{S}_0[i]$
  $Y_\ell \leftarrow (\boldsymbol{S}_1[i] \,\|\, \boldsymbol{T}_0[i])[1..|X|-k_1]$
  $Y_r \leftarrow (\boldsymbol{S}_1[i] \,\|\, \boldsymbol{T}_0[i])[|X|-k_1+1..|X|]$
  $\boldsymbol{C}[i] \leftarrow Y_\ell \,\|\, f(Y_r)$
$\mathsf{qr} \leftarrow \mathsf{true}$
return $\boldsymbol{C}$
```

Fig. 20. Games 2 and 3 for proof of Theorem 9.

```
Oracle LR($\boldsymbol{H}, \mathcal{M}$)                                   $\mathbf{G}_4(\mathcal{A}_1, k)$
$(\boldsymbol{M}_0, \boldsymbol{M}_1) \leftarrow\$ \mathcal{M}^{\mathsf{H}_1,\mathsf{G},\mathsf{H}_2}(1^k)$
$\boldsymbol{r} \leftarrow\$ \mathcal{R}^{\mathsf{H}_1,\mathsf{G},\mathsf{H}_2}(1^k)$
for $i \leftarrow 1$ to $v$ do
  $X \leftarrow \mathsf{pad}_n(\mathcal{E}_r(pk_r, \boldsymbol{H}[i], \boldsymbol{M}_b[i]\,;\boldsymbol{r}[i]))$
  $\boldsymbol{X}_\ell[i] \leftarrow X[1..|k_0|]$
  $\boldsymbol{X}_r[i] \leftarrow X[|k_0|+1..|X|]$
  $\boldsymbol{r}_1[i], \boldsymbol{r}_2[i] \leftarrow\$ \{0,1\}^{k_0}$
  $\boldsymbol{r}_G[i] \leftarrow\$ \{0,1\}^{n-k_0}$
  $a_1 \leftarrow T_1[pk_d \,\|\, \boldsymbol{X}_r[i]]$
  if $a_1 \neq \perp$ then $a_G \leftarrow T_G[pk_d \,\|\, (a_1 \oplus \boldsymbol{X}_\ell[i])]$
  else $a_G \leftarrow T_G[pk_d \,\|\, (\boldsymbol{r}_1[i] \oplus \boldsymbol{X}_\ell[i])]$
  if $a_G \neq \perp$ then $a_2 \leftarrow T_2[pk_d \,\|\, (a_G \oplus \boldsymbol{X}_r[i])]$
  else $a_2 \leftarrow T_2[pk_d \,\|\, (\boldsymbol{r}_G[i] \oplus \boldsymbol{X}_r[i])]$
  if $a_1 \neq \perp$ then
    if $a_G \neq \perp$ then $\mathsf{bad}_1 \leftarrow \mathsf{true}$
    else $\mathsf{bad}_{3,2} \leftarrow \mathsf{true}$; $\boldsymbol{r}_1[i] \leftarrow a_1$
  $\boldsymbol{S}_0[i] \leftarrow \boldsymbol{r}_1[i] \oplus \boldsymbol{X}_\ell[i]$
  if $a_G \neq \perp$ then
    if $a_1 \neq \perp$ then $\mathsf{bad}_1 \leftarrow \mathsf{true}$
    else $\mathsf{bad}_2 \leftarrow \mathsf{true}$
  $\boldsymbol{T}_0[i] \leftarrow \boldsymbol{r}_G[i] \oplus \boldsymbol{X}_r[i]$
  if $a_2 \neq \perp$ then
    if $a_1 = \perp \wedge a_G \neq \perp$ then
      $\mathsf{bad}_2 \leftarrow \mathsf{true}$
    else if $a_1 \neq \perp \wedge a_G = \perp$ then
      $\mathsf{bad}_{3,1} \leftarrow \mathsf{true}$; $\boldsymbol{r}_2[i] \leftarrow a_2$
    else $\mathsf{bad}_4 \leftarrow \mathsf{true}$; $\boldsymbol{r}_2[i] \leftarrow a_2$
  $\boldsymbol{S}_1[i] \leftarrow \boldsymbol{r}_2[i] \oplus \boldsymbol{S}_0[i]$
  $Y_\ell \leftarrow (\boldsymbol{S}_1[i] \,\|\, \boldsymbol{T}_0[i])[1..|X| - k_1]$
  $Y_r \leftarrow (\boldsymbol{S}_1[i] \,\|\, \boldsymbol{T}_0[i])[|X| - k_1 + 1..|X|]$
  $\boldsymbol{C}[i] \leftarrow Y_\ell \,\|\, f(Y_r)$
$\mathsf{qr} \leftarrow \mathsf{true}$
return $\boldsymbol{C}$
```

```
Oracle LR($\boldsymbol{H}, \mathcal{M}$)                                   $\mathbf{G}_5(\mathcal{A}_1, k)$
$(\boldsymbol{M}_0, \boldsymbol{M}_1) \leftarrow\$ \mathcal{M}^{\mathsf{H}_1,\mathsf{G},\mathsf{H}_2}(1^k)$; $\boldsymbol{r} \leftarrow\$ \mathcal{R}^{\mathsf{H}_1,\mathsf{G},\mathsf{H}_2}(1^k)$
for $i \leftarrow 1$ to $v$ do
  $X \leftarrow \mathsf{pad}_n(\mathcal{E}_r(pk_r, \boldsymbol{H}[i], \boldsymbol{M}_b[i]\,;\boldsymbol{r}[i]))$
  $\boldsymbol{X}_\ell[i] \leftarrow X[1..|k_0|]$
  $\boldsymbol{X}_r[i] \leftarrow X[|k_0|+1..|X|]$
  $\boldsymbol{r}_1[i], \boldsymbol{r}_2[i] \leftarrow\$ \{0,1\}^{k_0}$
  $\boldsymbol{r}_G[i] \leftarrow\$ \{0,1\}^{n-k_0}$
  $a_1 \leftarrow T_1[pk_d \,\|\, \boldsymbol{X}_r[i]]$
  if $a_1 \neq \perp$ then $a_G \leftarrow T_G[pk_d \,\|\, (a_1 \oplus \boldsymbol{X}_\ell[i])]$
  else $a_G \leftarrow T_G[pk_d \,\|\, (\boldsymbol{r}_1[i] \oplus \boldsymbol{X}_\ell[i])]$
  if $a_G \neq \perp$ then $a_2 \leftarrow T_2[pk_d \,\|\, (a_G \oplus \boldsymbol{X}_r[i])]$
  else $a_2 \leftarrow T_2[pk_d \,\|\, (\boldsymbol{r}_G[i] \oplus \boldsymbol{X}_r[i])]$
  if $a_1 \neq \perp$ then
    if $a_G \neq \perp$ then $\mathsf{bad}_1 \leftarrow \mathsf{true}$
    else $\mathsf{bad}_{3,2} \leftarrow \mathsf{true}$
  $\boldsymbol{S}_0[i] \leftarrow \boldsymbol{r}_1[i] \oplus \boldsymbol{X}_\ell[i]$
  $\boldsymbol{T}_0[i] \leftarrow \boldsymbol{r}_G[i] \oplus \boldsymbol{X}_r[i]$
  if $a_2 \neq \perp$ then
    if $a_1 \neq \perp \wedge a_G = \perp$ then
      $\mathsf{bad}_{3,1} \leftarrow \mathsf{true}$
    else $\mathsf{bad}_4 \leftarrow \mathsf{true}$; $\boldsymbol{r}_2[i] \leftarrow a_2$
  $\boldsymbol{S}_1[i] \leftarrow \boldsymbol{r}_2[i] \oplus \boldsymbol{S}_0[i]$
  $Y_\ell \leftarrow (\boldsymbol{S}_1[i] \,\|\, \boldsymbol{T}_0[i])[1..|X| - k_1]$
  $Y_r \leftarrow (\boldsymbol{S}_1[i] \,\|\, \boldsymbol{T}_0[i])[|X| - k_1 + 1..|X|]$
  $\boldsymbol{C}[i] \leftarrow Y_\ell \,\|\, f(Y_r)$
$\mathsf{qr} \leftarrow \mathsf{true}$
return $\boldsymbol{C}$
```

Fig. 21. Games 4 and 5 for proof of Theorem 9.

```
Oracle $\mathsf{H}_1(X)$
if $T_1[X] = \bot$ then $T_1[X] \leftarrow_\$ \{0,1\}^{k_0}$
if $\mathsf{qr} \wedge (\exists\, i \in [v])\; X = pk_d \,\|\, \boldsymbol{X}_r[i]$ then
    $T_1[X] \leftarrow \boldsymbol{r}_1[i]$
return $T_1[X]$

Oracle $\mathsf{G}(X)$
if $T_G[X] = \bot$ then $T_G[X] \leftarrow_\$ \{0,1\}^{n-k_0}$
if $\mathsf{qr} \wedge (\exists\, i \in [v])\; X = pk_d \,\|\, \boldsymbol{S}_0[i]$ then
    $T_G[X] \leftarrow \boldsymbol{r}_G[i]$
return $T_G[X]$

Oracle $\mathsf{H}_2(X)$
if $T_2[X] = \bot$ then $T_2[X] \leftarrow_\$ \{0,1\}^{k_0}$
if $\mathsf{qr} \wedge (\exists\, i \in [v])\; X = pk_d \,\|\, \boldsymbol{T}_0[i]$ then
    $\mathsf{bad}_6 \leftarrow \mathsf{true}$; $T_2[X] \leftarrow \boldsymbol{r}_2[i]$
return $T_2[X]$
```

```
Oracle $\mathbf{LR}(\boldsymbol{H}, \mathcal{M})$                    $\mathbf{G}_6(\mathcal{A}_1, k)$
$(\boldsymbol{M}_0, \boldsymbol{M}_1) \leftarrow_\$ \mathcal{M}^{\mathsf{H}_1,\mathsf{G},\mathsf{H}_2}(1^k)$
$\boldsymbol{r} \leftarrow_\$ \mathcal{R}^{\mathsf{H}_1,\mathsf{G},\mathsf{H}_2}(1^k)$
for $i \leftarrow 1$ to $v$ do
    $X \leftarrow \mathsf{pad}_n(\mathcal{E}_r(pk_r, \boldsymbol{H}[i], \boldsymbol{M}_b[i]\,;\boldsymbol{r}[i]))$
    $\boldsymbol{X}_\ell[i] \leftarrow X[1..|k_0|]$
    $\boldsymbol{X}_r[i] \leftarrow X[|k_0|+1..|X|]$
    $\boldsymbol{r}_1[i], \boldsymbol{r}_2[i] \leftarrow_\$ \{0,1\}^{k_0}$
    $\boldsymbol{r}_G[i] \leftarrow_\$ \{0,1\}^{n-k_0}$
    $\boldsymbol{S}_0[i] \leftarrow \boldsymbol{r}_1[i] \oplus \boldsymbol{X}_\ell[i]$
    $\boldsymbol{T}_0[i] \leftarrow \boldsymbol{r}_G[i] \oplus \boldsymbol{X}_r[i]$
    if $a_2 \neq \bot$ then
        $\mathsf{bad}_4 \leftarrow \mathsf{true}$
    $\boldsymbol{S}_1[i] \leftarrow \boldsymbol{r}_2[i] \oplus \boldsymbol{S}_0[i]$
    $Y_\ell \leftarrow (\boldsymbol{S}_1[i] \,\|\, \boldsymbol{T}_0[i])[1..|X|-k_1]$
    $Y_r \leftarrow (\boldsymbol{S}_1[i] \,\|\, \boldsymbol{T}_0[i])[|X|-k_1+1..|X|]$
    $\boldsymbol{C}[i] \leftarrow Y_\ell \,\|\, f(Y_r)$
$\mathsf{qr} \leftarrow \mathsf{true}$
return $\boldsymbol{C}$
```

```
$\mathcal{B}(1^k, f, y)$
$ctr \leftarrow 0$; $pk_d \leftarrow \langle f \rangle$
$(pk_r, sk_r) \leftarrow_\$ \mathcal{K}(1^k)$; $j \leftarrow_\$ [q_2]$; $w \leftarrow_\$ [v]$
$b' \leftarrow_\$ \mathcal{A}_1^{\mathbf{LR}',\mathsf{H}_1,\mathsf{G},\mathsf{H}_2}(1^k, \langle pk_r, pk_d \rangle)$
return $x$

Oracle $\mathbf{LR}'(\boldsymbol{H})$
for $i \leftarrow 1$ to $v$ do
    $Y_\ell \leftarrow_\$ \{0,1\}^{n-k_1}$; $Y_r \leftarrow_\$ \{0,1\}^{k_1}$
    if $i = w$ then $\boldsymbol{C}[i] \leftarrow Y_\ell \,\|\, y$
    else $\boldsymbol{C}[i] \leftarrow Y_\ell \,\|\, f(Y_r)$
return $\boldsymbol{C}$
```

```
Oracle $\mathsf{H}_1(X)$
if $T_1[X] = \bot$ then $T_1[X] \leftarrow_\$ \{0,1\}^{k_0}$
return $T_1[X]$

Oracle $\mathsf{G}(X)$
if $T_G[X] = \bot$ then $T_G[X] \leftarrow_\$ \{0,1\}^{n-k_0}$
return $T_G[X]$

Oracle $\mathsf{H}_2(X)$
if $T_2[X] = \bot$ then $T_2[X] \leftarrow_\$ \{0,1\}^{k_0}$
$ctr \leftarrow ctr + 1$
if $ctr = j \wedge X[1..|pk_d|] = pk_d$ then
    $x \leftarrow X[|pk_d|+1..|X|]$
return $T_2[X]$
```

Fig. 22. Game 6 and adversary $\mathcal{B}$ for proof of Theorem 9.