Hedging Public-Key Encryption in the Real World

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

[1] Georgia Institute of Technology, Atlanta, GA, USA
sasha@gatech.edu

[2] University of Florida, Gainesville, FL, USA

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.

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

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 [18,20,21,26,30,31]. Such failures have led to serious breaches of security in deployed cryptographic schemes [12,18,27,35]. 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. [27] 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 [18].

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 [5], 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)[1]. 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) [8], 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 [11,32,34,38]. It has also been used to construct deterministic encryption [4,13,34]. In fact, this trick of synthesizing randomness for encryption dates back (at least) to Fujisaki and Okamoto [24], 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 [9] specified in PKCS #1 v2.2

[1] To be precise, [5] refers to their constructions as REwH, and those are extensions of the EwH scheme from [4]. We use the name EwH for simplicity.

(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 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 Sect. 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 Sect. 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 punch-line 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 [5], 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 [5]. 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 [1,17,19,37]. 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) security.[2] 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 Sect. 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

[2] Consider the plaintext-recovery attack by Brown [15] on RSA-OAEP with public exponent $e = 3$. The attack exploits low entropy coins and is effective even if messages have high min-entropy.

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.

non-adaptive MMR security into adaptive MMR security only with the addition of an extra key-anonymity property (ANON); Bellare et al. [5] show this in the CPA setting, and we give an analogous result in the CCA setting (Theorem 1).

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 $\not\Rightarrow$ MM-CCA in general, which makes our positive result for RSA-OAEP non-trivial.

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. [5] 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 [4].

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 [4,5,13,34], 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 [4], 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 [5]), 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. [2] 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. [34] 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. [32] 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 [38] 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 [11] 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 [7,16] show that assuming the existence of indistinguishability obfuscation (iO), the random oracle in the EwH construction is uninstantiable. Finally, Hoang et al. [28] 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.

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.

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.)" [22].

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 [23] 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., by reading from `/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[3] 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.[4]) Second, the documentation does not specify how the state is updated, except that if `entropy` is equal to `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 [5, 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

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

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 [10]. 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 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.

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 [5]. 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 [5, Sect. 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 [6]), 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 [5] (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 Fig. 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 Fig. 2. We say that PKEAD is IND-CCA secure if for every PT ("polynomial-time") adversary $\mathcal{A}$, the function

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

$\mathbf{Adv}^{\text{ind-cca}}_{\mathsf{PKEAD}}(\mathcal{A}, \cdot)$ 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 [5] to deal with associated data and chosen-ciphertext attacks.

Consider the MMR-CCA experiment defined in Fig. 2 associated to PKEAD, adversary $\mathcal{A}$, and security parameter k. The output of the **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 **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.

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 [5], building on [4]) 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.

Min-entropy requirements. Just as in prior work [4,5], 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.

4.3 ANON Security

Bellare et al. [5] 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 Fig. 3.

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

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

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

```
Oracle $\mathbf{Dec}_b(H, C)$:
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 [5], lifted to the CCA setting.

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 [3], 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 [5], which in turn lifts [3] 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 [5, Theorem 5.2] to the CCA setting.

Theorem 1 (MMR1+ANON-CCA $\implies$ 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*

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

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 [5, Theorem 5.2] that takes the decryption oracle into account; we refer the reader to the full version of this paper for the details [14]. 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 Fig. 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 sequence of coins. This allows the adversary to mount the key-dependent distribution attack identified by [5] when the adversary is given the public key. (Indeed, this kind of attack is effective whenever the randomness source has low min-entropy. 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

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:** $\mathsf{Enc}_{pk}^{H}(M; r) = \mathcal{E}_{pk}^{H}(M; r) \,\|\, \mathsf{H}(pk)$, where $\mathcal{E}$ is {MM,MMR}-CPA and H a random oracle; **CE2:** $\mathsf{Enc}_{pk}^{H}(M; r) = M$; **CE3:** EME-OAEP (see Sect. 6.1); **CE4:** $\mathsf{Enc}_{pk}^{H}(M; r \,\|\, b) = \mathcal{E}_{pk}^{H}(M; r) \,\|\, (b \oplus M[1])$, 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.

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 [27]. 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 [31], or the "boot-time entropy hole" described in [27], 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 Fig. 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*

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

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.*

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. The full proof can be found in the full version of this paper [14].

Finally, we exhibit a scheme that is MM-CPA, but *not* MMR-CPA in Sect. 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 $\Pi^{\text{mm-cca}}_{0,\varphi,1} \subseteq$ 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) $|S| = v 2^{\mu_1}$; (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 $X_0 \neq Y_0$ and $X_1 \neq Y_1$. 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

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

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

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

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

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

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. [5]. We show that if the deterministic scheme is instantiated specifically by RSA-DOAEP [4], which can be done via high-level API calls to hash functions and RSA, then PtD achieves MM+IND-CCA, and RtD achieves MM+IND-CPA. We also suggest specific conditions under which RtD would be MMR+IND-CCA, extending prior work [5].

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[5] $n(\cdot)$ 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

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

is a negligible function of k.

We will also use the stronger security notion of *partial-domain one-wayness* formalized by Fujisaki et al. [25], 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:

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

6.1 EME-OAEP

We first look at RSA-OAEP [9], the only provably-secure PKE scheme available in OpenSSL, and indeed most libraries.[6] It is known to be IND-CCA secure assuming that the underlying trapdoor permutation is POWF secure, or under the RSA assumption [25,36].

[5] For example, the input length might be the number of modulus bits in RSA.
[6] Some implement ElGamal or hybrid encryption schemes as well.

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

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

$\mathsf{Dec}^H_{sk}(C)$
    $\langle f^{-1}\rangle \leftarrow sk$; $P \leftarrow f^{-1}(C)$
    if $|P| \neq n$ then return $\bot$
    $X_1 \,\|\, Y_1 \,\|\, [z] \leftarrow P$   # $|Y_1| = \rho$
    $Y_0 \leftarrow Y_1 \oplus \mathsf{H}_2(X_1)$
    $X_0 \leftarrow X_1 \oplus \mathsf{G}(Y_0)$
    $PM \,\|\, T \leftarrow X_0$   # $|T| = \tau$
    if $\mathsf{H}_1(H) \neq T$ then return $\bot$
    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 $\mathsf{pad} : \{0,1\}^* \to \{0,1\}^{m-\tau} \cup \{\bot\}$ 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 $\mathsf{pad}(M) = \bot$ otherwise. Define its inverse $\mathsf{unpad} : \{0,1\}^{m-\tau} \to \{0,1\}^* \cup \{\bot\}$ in the natural way.

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-EME-OAEP specified in Fig. 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.[7] 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.[8]

F-EME-OAEP IS NOT MMR-CPA. This scheme is not MMR-CPA secure, due to an attack by Brown [15] on RSA-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 RSA-EME-OAEP.

F-EME-OAEP IS MM-CCA. We prove the scheme does achieve our new notion. The standard cites the result of [25] 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-EME-OAEP is MM-CCA secure with respect to high min-entropy coins sources, assuming that F is POWF

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

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

secure. By [25, Lemma 4.2], instantiating the trapdoor with RSA is secure assuming only that RSA is OWF secure.

Theorem 5 (F-EME-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 $\rho(k)+\tau(k)+16 \leq n(k)$ for every $k \in \mathbb{N}$. Let $m(k) = n(k)-\rho(k)-8$. Let PKEAD = F-EME-OAEP as defined in Fig. 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*

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

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

The proof appears in the full version [14]. 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 Sect. 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 RSA-EME-OAEP barring catastrophic failure of the (P)RNG. If the adversary is able to guess the coins used, then there is an attack [15], and so the Dual EC DRBG attack [18], for example, completely breaks the security of RSA-EME-OAEP. Even cases where the coins still have *some* entropy [31] we consider insecure in an asymptotic sense, since an adversary can guess the coins with non-negligible probability.

MMR does not imply MM security. Since F-EME-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 AEAD = (Kgen, Enc, Dec). The randomized *key generation* algorithm Kgen samples a key K from a finite, non-empty set $\mathcal{K}$

```
$\underline{\mathsf{Kgen}(1^k)}$:
    $(f, f^{-1}) \leftarrow_\$ F(1^k)$
    $R \leftarrow_\$ \{0,1\}^r$
    return $(f \,\|\, R, f^{-1})$
```

```
$\underline{\mathsf{Enc}_{pk}(M, H)}$
    $X \leftarrow_\$ \{0,1\}^{\rho(k)}$
    $K_P \leftarrow \mathsf{H}_1(\langle f \,\|\, R, H, M, X\rangle)$
    $C_1 \leftarrow f(K_P)$
    $K \leftarrow \mathsf{H}_2(\langle f \,\|\, R, H, K_P\rangle)$
    $\tilde{H} \leftarrow \langle H, C_1\rangle$
    $N \leftarrow \mathsf{extract}(\tilde{H})$
    $C_2 \leftarrow \mathsf{AEAD.Enc}(K, N, \tilde{H}, M)$
    return $C_1 \,\|\, C_2$
```

```
$\underline{\mathsf{Dec}_{sk}(H, C_1 \,\|\, C_2)}$
    $K_P \leftarrow f^{-1}(C_1)$
    $K \leftarrow \mathsf{H}_2(\langle f \,\|\, R, H, K_P\rangle)$
    $\tilde{H} \leftarrow \langle H, C_1\rangle$
    $N \leftarrow \mathsf{extract}(\tilde{H})$
    $M \leftarrow \mathsf{AEAD.Dec}(K, N, \tilde{H}, C_2)$
    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.

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 $\mathbf{Adv}^{\mathrm{auth}}_{\mathsf{AEAD}}(\mathcal{A}) = \Pr[\mathbf{Exp}^{\mathrm{auth}}_{\mathsf{AEAD}}(\mathcal{A})]$.

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 Fig. 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 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 $\mathsf{PKEAD} = \mathsf{HE}[\mathrm{F}, \mathsf{AEAD}]$ as defined in Fig. 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 $\mathbf{LR}$ oracle, q_d queries to its $\mathbf{Dec}$ 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*

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

– *(IND-CCA) Let $\mathcal{A}$ be an adversary attacking PKEAD and making q queries to its $\mathbf{LR}$ oracle, q_d queries to its $\mathbf{Dec}$ 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*

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

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 the full version of this paper [14]. 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.

6.3 Generic Constructions

We describe two black-box constructions of [5], 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 $\{0,1\}^* \cup \{\bot\}$. Decryption deterministically maps the secret key sk and a string C to an element of $\{0,1\}^* \cup \{\bot\}$. 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 $\mathcal{E}_{pk}(M) \neq \bot$ 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 [4]. 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 Sect. 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 [5] for a precise definition.)

LOSSY AND ALL-BUT-ONE TRAPDOOR FUNCTIONS. LTDFs were first described by Peikert and Waters [33]. Informally, an *LTDF generator* F is a probabilistic algorithm that on input 1^k and $b \in \{0,1\}$ outputs a pair of strings (s,t) such that s encodes a function f. If $b = 1$, then function f is injective, and t encodes a function f^{-1} giving its inverse; otherwise, the image of f is significantly smaller than the injective mode (i.e., $b = 1$). The generator is secure if no reasonable adversary, given s, can distinguish 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, [33] introduce ABO ("all-but-one") TDFs as a richer abstraction. Instead of having an injective and lossy mode, an ABO TDF has a *set of modes*, one of which is lossy. Here, security demands that every pair of modes are computationally indistinguishable. Both primitives have been constructed from a number of hardness assumptions: For example, the Φ-hiding assumption for RSA [29] and LWE ("learning with errors") for lattices [33]. A universal LTDF is given by Boldyreva, Fehr, and O'Neill [13] 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

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

```
$\mathsf{PtD}[\Pi_d].\mathsf{Enc}_{pk}^{H}(M)$
    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 $\mathcal{E}_d(pk, H \,\|\, PM)$
```

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

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

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

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

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

```
$F\text{-}\mathsf{DOAEP}.\mathcal{E}_{pk}(X)$
    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 $Y_\ell \,\|\, f(Y_r)$
```

```
$F\text{-}\mathsf{DOAEP}.\mathcal{D}_{sk}(Y)$
    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.

top panel of Fig. 7. This is the same as the construction proposed by [5], 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 Fig. 7. Bellare et al. [5, Theorem 6.3] prove this construction is MMR-CPA if Π is MMR-CPA, and IND-CPA if Π is a u-LTDF. [9] 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 [13, Theorem 5.1], and ANON-CPA secure for block-sources [5, Theorem 5.3]. Thus, the scheme $\mathsf{PtD}[\Pi]$ is MMR-hedged secure (for block-sources) against chosen-distribution attacks as long

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

as Π is a u-LTDF. Note that universal-inducing property is not essential; see [5, Sect. 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 [33] 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 [13] 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.[10] 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 Fig. 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*

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

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

[10] In another direction, [34] consider novel notions of LTDFs for their adaptive CCA setting.

Let us explain this claim a bit. (The proof is in the full version [14].) 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 [4, 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 [4, Lemma A.1]. (This is closely related to the POWF notion of [25].) 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 Sect. 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 Fig. 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 [5, 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 [5], 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 [23] and OAEP [9].

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 Fig. 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 [5, Theorem 6.2]; The details appear in the full version of this paper [14]. 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 [4]. (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* MMR+IND-CPA).** *Let F be a trapdoor permutation generator with randomness length $k_1(\cdot)$. Let F-DOAEP be the deterministic scheme defined in Fig. 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 Fig. 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 $\mathbf{LR}$ 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 RSA-DOAEP is PRIV secure [4, 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 Fig. 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 the full version of this paper for the proof [14].

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

1. Abdalla, M., Benhamouda, F., Pointcheval, D.: Public-key encryption indistinguishable under plaintext-checkable attacks. IET Inf. Secur. **10**(6), 288–303 (2016)
2. Acar, Y., Backes, M., Fahl, S., Garfinkel, S., Kim, D., Mazurek, M.L., Stransky, C.: Comparing the usability of cryptographic apis. In: Proceedings of the 38th IEEE Symposium on Security and Privacy (2017)
3. Bellare, M., Boldyreva, A., Desai, A., Pointcheval, D.: Key-privacy in public-key encryption. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 566–582. Springer, Heidelberg (2001). doi:10.1007/3-540-45682-1_33

4. Bellare, M., Boldyreva, A., O'Neill, A.: Deterministic and efficiently searchable encryption. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 535–552. Springer, Heidelberg (2007). doi:10.1007/978-3-540-74143-5_30
5. Bellare, M., Brakerski, Z., Naor, M., Ristenpart, T., Segev, G., Shacham, H., Yilek, S.: Hedged public-key encryption: how to protect against bad randomness. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 232–249. Springer, Heidelberg (2009). doi:10.1007/978-3-642-10366-7_14
6. Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations among notions of security for public-key encryption schemes. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 26–45. Springer, Heidelberg (1998). doi:10.1007/BFb0055718
7. Bellare, M., Hoang, V.T.: Resisting randomness subversion: fast deterministic and hedged public-key encryption in the standard model. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 627–656. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46803-6_21
8. Bellare, M., Rogaway, P.: 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)
9. Bellare, M., Rogaway, P.: Optimal asymmetric encryption. In: Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 92–111. Springer, Heidelberg (1995). doi:10.1007/BFb0053428
10. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 409–426. Springer, Heidelberg (2006). doi:10.1007/11761679_25
11. Bellare, M., Tackmann, B.: Nonce-based cryptography: retaining security when randomness fails. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9665, pp. 729–757. Springer, Heidelberg (2016). doi:10.1007/978-3-662-49890-3_28
12. Bernstein, D.J., Chang, Y.-A., Cheng, C.-M., Chou, L.-P., Heninger, N., Lange, T., Someren, N.: Factoring RSA keys from certified smart cards: Coppersmith in the wild. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013. LNCS, vol. 8270, pp. 341–360. Springer, Heidelberg (2013). doi:10.1007/978-3-642-42045-0_18
13. Boldyreva, A., Fehr, S., O'Neill, A.: On notions of security for deterministic encryption, and efficient constructions without random oracles. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 335–359. Springer, Heidelberg (2008). doi:10.1007/978-3-540-85174-5_19
14. Boldyreva, A., Patton, C., Shrimpton, T.: Hedging public-key encryption in the real world. Cryptology ePrint Archive, Report 2017/510 (2017). http://eprint.iacr.org/2017/510
15. Brown, D.R.L.: A weak-randomizer attack on RSA-OAEP with $e = 3$. Cryptology ePrint Archive, Report 2005/189 (2005). http://eprint.iacr.org/2005/189
16. Brzuska, C., Farshim, P., Mittelbach, A.: Random-oracle uninstantiability from indistinguishability obfuscation. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015. LNCS, vol. 9015, pp. 428–455. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46497-7_17
17. Camenisch, J., Chandran, N., Shoup, V.: A public key encryption scheme secure against key dependent chosen plaintext and adaptive chosen ciphertext attacks. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 351–368. Springer, Heidelberg (2009). doi:10.1007/978-3-642-01001-9_20

18. Checkoway, S., Maskiewicz, J., Garman, C., Fried, J., Cohney, S., Green, M., Heninger, N., Weinmann, R.P., Rescorla, E., Shacham, H.: A systematic analysis of the Juniper Dual EC incident. In: CCS 2016 Proceedings of the 23rd ACM Conference on Computer and Communications Security (2016)
19. Dodis, Y., Katz, J.: Chosen-ciphertext security of multiple encryption. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 188–209. Springer, Heidelberg (2005). doi:10.1007/978-3-540-30576-7_11
20. Dodis, Y., Pointcheval, D., Ruhault, S., Vergniaud, D., Wichs, D.: 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)
21. Dorrendorf, L., Gutterman, Z., Pinkas, B.: Cryptanalysis of the random number generator of the windows operating system. ACM Trans. Inf. Syst. Secur. **13**(1), 10 (2009)
22. Duong, T., Kasper, E., Nguyen, Q.: Scaling Crypto Testing with Project Wycheproof. https://www.cs.bris.ac.uk/Research/CryptographySecurity/RWC/2017/thai.duong.pdf
23. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985). doi:10.1007/3-540-39568-7_2
24. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (1999). doi:10.1007/3-540-48405-1_34
25. Fujisaki, E., Okamoto, T., Pointcheval, D., Stern, J.: RSA-OAEP is secure under the RSA assumption. J. Cryptol. **17**(2), 81–104 (2004)
26. Gutterman, Z., Malkhi, D.: Hold your sessions: an attack on java session-id generation. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 44–57. Springer, Heidelberg (2005). doi:10.1007/978-3-540-30574-3_5
27. Heninger, N., Durumeric, Z., Wustrow, E., Halderman, J.A.: 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)
28. Hoang, V.T., Katz, J., O'Neill, A., Zaheri, M.: Selective-opening security in the presence of randomness failures. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10032, pp. 278–306. Springer, Heidelberg (2016). doi:10.1007/978-3-662-53890-6_10
29. Kiltz, E., O'Neill, A., Smith, A.: Instantiability of RSA-OAEP under chosen-plaintext attack. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 295–313. Springer, Heidelberg (2010). doi:10.1007/978-3-642-14623-7_16
30. Michaelis, K., Meyer, C., Schwenk, J.: Randomly failed! the state of randomness in current java implementations. In: Dawson, E. (ed.) CT-RSA 2013. LNCS, vol. 7779, pp. 129–144. Springer, Heidelberg (2013). doi:10.1007/978-3-642-36095-4_9
31. Mueller, M.: Debian OpenSSL predictable PRNG (2008). https://www.exploit-db.com/exploits/5622/
32. Paterson, K.G., Schuldt, J.C.N., Sibborn, D.L.: Related randomness attacks for public key encryption. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 465–482. Springer, Heidelberg (2014). doi:10.1007/978-3-642-54631-0_27
33. Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. In: STOC 2008 Proceedings of the 40th Annual ACM Symposium on Theory of Computing (2008)

34. Raghunathan, A., Segev, G., Vadhan, S.: Deterministic public-key encryption for adaptively chosen plaintext distributions. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 93–110. Springer, Heidelberg (2013). doi:10.1007/978-3-642-38348-9_6
35. Ristenpart, T., Yilek, S.: 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)
36. Shoup, V.: OAEP reconsidered. J. Cryptol. **15**(4), 223–249 (2002)
37. Shoup, V.: ISO18033-2: An emerging standard for public-key encryption (final committee draft) (2004). http://shoup.net/iso
38. Yilek, S.: Resettable public-key encryption: how to encrypt on a virtual machine. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 41–56. Springer, Heidelberg (2010). doi:10.1007/978-3-642-11925-5_4