# The Local Forking Lemma and its Application to Deterministic Encryption

Mihir Bellare[1], Wei Dai[1], and Lucy Li[2]

[1] University of California, San Diego, {mihir, weidai}@eng.ucsd.edu
[2] Cornell Univeristy, lucy@cs.cornell.edu

**Abstract.** We bypass impossibility results for the deterministic encryption of public-key-dependent messages, showing that, in this setting, the classical Encrypt-with-Hash scheme provides message-recovery security, across a broad range of message distributions. The proof relies on a new variant of the forking lemma in which the random oracle is reprogrammed on just a single fork point rather than on all points past the fork.

## 1 Introduction

Deterministic encryption. In a scheme DE for Deterministic Public-Key Encryption (D-PKE) [2], the encryption algorithm DE.Enc takes public encryption key $ek$ and message $m$ to deterministically return a ciphertext $c$. The standard privacy goal is most easily understood as the same as for randomized public-key encryption —IND-CPA, asking for indistinguishability of encryptions of different messages— but with two restrictions: (1) That messages not depend on the public key, and (2) that messages be unpredictable, meaning have high min entropy. We will use the IND formalism of [5], shown by the latter to be equivalent to the PRIV formalization of [2] as well as to several other formalizations. A canonical and practical construction is EwH (Encrypt with Hash) [2]. It encrypts message $m$ under a (any) randomized IND-CPA scheme RE with the coins set to a hash of $ek\|m$, and is proven IND-secure if the hash function is a random oracle [2]. Further schemes and considerations can be found in [13, 14, 27, 30, 19, 6, 20].

Why D-PKE? Determinism allows sorting of ciphertexts, enabling fast search on encrypted data, the motivating application in BBO's introduction of D-PKE [2]. Determinism also closes the door to vulnerabilities arising from poor randomness [15, 28]. Understood to be a threat already when its causes were inadvertent system errors [31], poor randomness is now even more a threat when we see that it can be intentional, arising from the subversion of RNGs happening as part of mass-surveillance activities [11].

Narrowing the gap. We benefit, in light of the above motivations for D-PKE, from the latter providing privacy as close to IND-CPA as possible. We can't expect of course to entirely close the gap —no D-PKE scheme can achieve IND-CPA— but we'll narrow it. Our target will be the first of the two limitations of IND noted above, namely that it guarantees no privacy when messages depend on the

public key. In particular, for all we know, in this case, one could recover the entire message from an $\mathsf{EwH}$-ciphertext. This is the gap we will close, showing $\mathsf{EwH}$ message recovery is not possible across a broad range of public-key-dependent message distributions. We'll explain how this bypasses, rather than contradicts, prior impossibility results that have inhibited progress on the question, while also contributing new, more fine-grained impossibility results to indicate that our own possibility results will not extend much beyond the message distributions for which we establish them. Underlying our possibility result is a new variant of the forking lemma [29, 7, 1], that we call the Local Forking Lemma, of independent interest. We now look at all this in more detail.

Prior work. Given that the public key is, as the name indicates, public, messages depending on it are a possibility in practice, and IND-CPA provides privacy even for such messages. But, for D-PKE, the literature says that security for public-key-dependent messages is impossible [2, 5]. The argument supporting this claim is that the following attack violates IND-security of *any* D-PKE scheme $\mathsf{DE}$. In its message finding stage, the adversary, given the public key $ek$, picks $m_0, m_1$ at random —of length, say, equal to the security parameter— subject to the constraint that the first bits of $h_0 \leftarrow \textsc{Hash}(\mathsf{DE.Enc}(ek, m_0))$ and $h_1 \leftarrow \textsc{Hash}(\mathsf{DE.Enc}(ek, m_1))$ are $0, 1$, respectively, where $\textsc{Hash}$ is a random oracle. The messages $m_0, m_1$ are unpredictable, but given a ciphertext $c \leftarrow \mathsf{DE.Enc}(ek, m_b)$ encrypting $m_b$, the adversary can determine $b$ as the first bit of the hash $\textsc{Hash}(c)$ of the ciphertext.

That IND cannot be achieved for public-key dependent messages doesn't mean no security is possible in this setting; perhaps guarantees can be provided under some other, meaningful metric (definition) of security X. Raghunathan, Segev and Vadhan (RSV) [30] were the first to pursue this, making a choice of X that we'll refer to as PDIND. In X=PDIND, security is parameterized by the number $N(\cdot)$ of (public-key dependent) distributions from which the message may be drawn. RSV [30] show that, if one first fixes an upper bound $N(\cdot) = 2^{p(\cdot)}$ on the number of allowed message distributions, then one can build a PDIND secure D-PKE scheme, with the scheme and its parameters depending on $N(\cdot)$. While theoretically interesting, this result has limitations from a practical perspective. The scheme is expensive, with key size and computation time growing polynomially with $p$, and this is inherent. Security is fragile: If the number of message distributions exceeds the bound $N(\cdot)$, security may —and in some of their schemes, will— fail. There is difficulty of use: it is not clear how a designer or implementer can, with confidence, pick $N(\cdot)$ a priori, but they must have $N(\cdot)$ in hand to build the scheme.

PDMR security. Our target is a simple, meaningful security guarantee (when deterministically encrypting public-key dependent messages) that we can establish for *practical* schemes. We reach this by making a different choice of X above. We formalize and target X=PDMR, *message recovery* security for public-key-dependent messages. The definition, in Section 4, considers a *source* $S$ that, given the public key $ek$ and access to the random oracle $\textsc{Hash}$, returns a sequence of

unpredictable messages. Encryptions under $ek$ of these messages are then provided to the adversary $A$, who, continuing to have $ek$ and access to HASH, must, to win, recover (in full) one of the messages. Unlike PDIND [30], there is no a priori restriction on the number of message distributions (here, sources).

One might object that message recovery security is a weak security guarantee, in response to which we note the following. *First*, in practice adversaries benefit more by recovering the full message from a ciphertext than by merely distinguishing the encryptions of two messages. So, even when distinguishing attacks are possible, a scheme preventing message recovery can add significant security. *Second*, right now, practical schemes like $\mathsf{EwH}$ are not proven to provide *any* security for public-key dependent messages, so if we can show PDMR is present, we have improved security guarantees without increasing cost. *Third*, in providing PDMR, we will insist that IND be maintained, so that overall security only goes up, not down. In other words, for messages not depending on the public key, we continue to provide the guarantee that is standard and viewed as best possible (IND), supplementing this with a meaningful guarantee (PDMR) for messages that do depend on the public key.

It is useful to define $n(\cdot)$-PDMR security as PDMR security for sources that output $n$ messages. We will establish PDMR first for $n = 1$ and then boost to more messages.

One-message PDMR security of $\mathsf{EwH}$. The core possibility result of this paper is that $\mathsf{EwH}$ is 1-PDMR secure, meaning provides message-recovery security for the encryption of one unpredictable message *even when the latter depends arbitrarily on the public key*. The underlying randomized public-key encryption scheme $\mathsf{RE}$ is assumed, only and correspondingly, to itself provide security against message-recovery. (This is implied by IND-CPA and hence true for $\mathsf{EwH}$ [2], but strictly weaker.) The hash function HASH continues, as in [2], to be modeled as a random oracle.

The proof requires new techniques. Let $m$ denote the challenge message produced by the source, and let $c_1 \leftarrow \mathsf{RE.Enc}(ek, m; r_1)$ where $r_1 \leftarrow \text{HASH}(ek\|m)$. The approach of [2] would replace $c_1$ with a ciphertext $c_0 \leftarrow \mathsf{RE.Enc}(ek, m; r_0)$ for random $r_0$, allowing a reduction to the assumed message-recovery security of $\mathsf{RE}$. This requires that neither the source nor the adversary make query $ek\|m$ to HASH, for otherwise they can differentiate $c_0$ from $c_1$. But this in turn requires that the source not have $ek$. Indeed, in our setting, where it does have $ek$, it *can* query $ek\|m$ to HASH, and we must assume that it does so. The prior argument now breaks down entirely and it is not clear how to do the reduction. We obtain our result, instead, via a novel rewinding argument. Two executions are forked at the crucial hash query, one corresponding to response $r_1$ and the other to response $r_0$, but with a twist. In the classical rewinding technique [29, 7], *all* answers to random-oracle queries after the fork are random and independent in the two forks. This fails to work in our case. Instead we are able to re-program the random oracle at just one point in the rewinding and argue that the two executions both result in correct guesses by the adversary.

The analysis relies on what we call the Local Forking lemma, a (new) variant of the forking lemmas of [29, 7, 1] that we give and prove. As with the General Forking Lemma of BN [7], our Local Forking Lemma is a purely probabilistic result, knowing or saying nothing about encryption. Handing off to our Local Forking Lemma the core probabilistic analysis in the above-discussed proof of 1-PDMR security not only makes the latter more modular but allows an extension to security against chosen-ciphertext attacks.

Many-message PDMR security of EwH. We show that EwH provides PDMR security for all sources (distributions on message sequences) that are what we call resampling indistinguishible (RI). Very roughly —the formal definition is in Section 5— RI asks that different messages in the sequence, although all allowed to depend on the public key in different ways, are themselves almost independently distributed.

Our first step is a general result showing that if a D-PKE scheme is 1-PDMR then it provides PDMR for *any* RI source. That is, once we have PDMR security when encrypting just one, single message, we also have it when encrypting *any* polynomial number of RI messages. This is a general result, holding for *any* D-PKE scheme. An interesting element of this result is that the public-key dependence of messages is a plus, exploited crucially in the proof.

To put this in context, for IND, security for one message does not, in general —that is, for arbitrary message distributions— imply security for multiple messages [2]. It has been shown to do so for particular message distributions, namely block sources, by Boldyreva, Fehr and O'Neill [13]. But they do not consider public-key-dependent messages, and block sources and RI distributions are not the same.

That EwH provides PDMR security for all RI sources now of course follows directly from the general reduction just mentioned and our above-discussed result establishing 1-PDMR security of EwH.

That these results are for EwH rather than some other scheme is important for two reasons. The first is that EwH is efficient and practical. The second is that we know that EwH already achieves IND for messages that do not depend on the public key [2]. As discussed above it is important that PDMR be provided while maintaining IND so that we augment, not reduce, existing guarantees.

CCA too. All the above considered security under chosen-plaintext attack (CPA). This is certainly the first and foremost goal, but one can ask also about security against chosen-ciphertext attack (CCA), particularly if our quest is parity (to the best extent possible) with randomized encryption, where motivated by applications [12], efficient IND-CCA schemes have been sought and provided [9, 18, 17, 25, 23, 21].

Our results extend to CCAs. Namely, we show that, under chosen-ciphertext attack, EwH continues to provide 1-PPDMR, and PDMR for RI sources, assuming the underlying randomized public-key encryption scheme itself provides message recovery under CCA, which is implied by IND-CCA. Put another way, EwH promotes message-recovery security of RE to message-recovery security of

the constructed DE, in both the CPA and the CCA cases, for public-key dependent messages produced by RI sources. In the body of the paper, we give unified definitions and a single, unified result that cover both CCA and CPA by viewing the latter as the special case of the former in which adversaries make no decryption queries, exploiting our Local Forking lemma to provide a modular proof.

Impossibility results. Our possibility results show that PDMR security is achievable when messages in the sequence are somewhat independent of each other, formalized as RI. We complement these possibility results with negative ones, showing that, when messages in a sequence are closely related, PDMR security is not possible. Section 6 gives attacks to show that PDMR security can be violated even when encrypting just two, closely related messages, even though both messages are unpredictable. This is true for *any* D-PKE scheme. These attacks are novel; the above-mentioned attacks understood in the literature violate indistinguishability security for public-key-dependent messages, but do not recover messages and thus, unlike ours, do not violate PDMR. We believe that a contribution here is not just to give these attacks, but with rigorous and formal analyses (Theorems 4 and 5), which is unusual in the literature. The proof of unpredictability in Theorem 5 relies on techniques from the proof of the Leftover Hash Lemma [24].

Discussion and further directions. It is interesting to note that Goldwasser and Micali's original definition of semantic security for public-key encryption [22] only required privacy for messages not depending on the public key. This was pointed out by Micali, Rackoff and Sloan (MRS) [26], who strengthened the definition in this regard. (In their terminology, this corresponds to three pass versus one pass notions.) Modern definitions of semantic security (IND-CPA) [4, 16] accordingly ask for privacy even for messages that depend on the public key, and modern public-key encryption schemes provide this privacy. Our work continues the quest, started by RSV [30], to bring D-PKE to parity as much as possible in this regard.

There is a great deal of work on D-PKE including many schemes without random oracles [13, 14, 27, 30, 19, 6, 20]. A direction for future work is to assess whether these schemes provide PDMR security, or give new schemes without random oracles that provide both IND and PDMR security.

The full and most current version of this paper is available as [3].

## 2 Preliminaries

Notation and terminology. By $\lambda \in \mathbb{N}$ we denote the security parameter and by $1^\lambda$ its unary representation. We denote the number of coordinates of a vector $\mathbf{x}$ by $|\mathbf{x}|$, the length of a string $x \in \{0,1\}^*$ by $|x|$ and the size of a set $S$ by $|S|$. If $x$ is a string then $x[i]$ is its $i$-th bit. Algorithms are randomized unless otherwise indicated. Running time is worst case. "PT" stands for "polynomial-time," whether for randomized algorithms or deterministic ones. For integers

$a \le b$ we let $[a..b] = \{a, a+1, \ldots, b\}$. We let $y \leftarrow A^{\mathsf{O}_1,\cdots}(x_1, \ldots; r)$ denote executing algorithm $A$ on inputs $x_1, \ldots$ and coins $r$ with access to oracles $\mathsf{O}_1, \ldots$ and letting $y$ be the result. We let $y \leftarrow\!\!\$\ A^{\mathsf{O}_1,\cdots}(x_1, \ldots)$ be the resulting of picking $r$ at random and letting $y \leftarrow A^{\mathsf{O}_1,\cdots}(x_1, \ldots; r)$. We let $[A^{\mathsf{O}_1,\cdots}(x_1, \ldots)]$ denote the set of all possible outputs of $A$ when invoked with inputs $x_1, \ldots$ and oracles $\mathsf{O}_1, \ldots$. We use $q_A^{\mathsf{O}_i}$ to denote the number of queries that $A$ makes to $\mathsf{O}_i$ in the worst case. We recall that a function $f\colon\ \mathbb{N} \to \mathbb{R}$ is negligible if for every positive polynomial $p$, there exists $n_p \in \mathbb{N}$ such that $f(n) < 1/p(n)$ for all $n > n_p$. An adversary is an algorithm or a tuple of algorithms. The running time of a tuple of algorithms is defined as the sum of the individual running times. We use $t_A$ to denote the running time of an adversary $A$.

Games. We use the code based game playing framework of [10]. (See Fig. 4 for an example.) By $\mathrm{G} \Rightarrow y$ we denote the event that the execution of game G results in output $y$, the game output being what is returned by the game. We write $\Pr[\mathrm{G}]$ as shorthand for $\Pr[\mathrm{G} \Rightarrow \mathsf{true}]$, the probability that the game returns $\mathsf{true}$.

Random Oracle Model (ROM). In the ROM [8], we give parties a random oracle HASH that on input a string $x \in \{0,1\}^*$ returns a an output $y$ that is (conceptually at least) a random, infinite string. The caller will then read a prefix of $y$, of any length it wants, and be charged, in terms of computation, an amount proportional only to the number of bits read.

Let $\mathbf{T}$ denote the set of all functions $T\colon\ \{0,1\}^* \to \{0,1\}^\infty$. Then, mathematically, a random oracle HASH is a function drawn at random from $\mathbf{T}$. We view each $T \in \mathbf{T}$ as a table so that values in it can be reprogrammed, and thus may write $T[\cdot]$ in place of $T(\cdot)$. HASH could be a procedure in games, for example in Fig. 3, where return values are sampled lazily as they are needed. Alternatively, we also sample the table $T$ that describes HASH uniformly at random from $\mathbf{T}$ at the beginning of the game (and write $T$ in place of HASH), for example in Fig. 1. We note that the above two ways of implementing the random oracle HASH are equivalent.

It is sometimes useful to give parties a variable output length random oracle. This takes two inputs, $x \in \{0,1\}^*$ and $\ell \in \mathbb{N}$, and returns a random $\ell$-bit string, and, even for a fixed $x$, the outputs for different lengths $\ell$ must be independent. We can implement such a variable output length RO in our model above, and now discuss how. First, what does *not* work is to query $x$ and take the $\ell$-bit prefix of the infinite-length string returned, since in this case the result for $x, \ell$ is a prefix of the result for $x, \ell'$ whenever $\ell' > \ell$, and so the two are not independent as required. However, one can first fix an efficient injective encoding of the form $\{0,1\}^* \times \mathbb{N} \to \{0,1\}^*$. Then, a query of the form $x, \ell$ to a variable-length RO can be simulated by quering encoding of the pair $(x, \ell)$ to our single-input random oracle HASH. With this understood, we will work in our model above.

| GAME $\mathrm{G}^{\mathrm{single}}_{\mathrm{SAMP},F}$ | GAME $\mathrm{G}^{\mathrm{double}}_{\mathrm{SAMP},F}$ |
|---|---|
| $\pi \leftarrow\!\!\$\ \mathrm{SAMP}()$ | $\pi \leftarrow\!\!\$\ \mathrm{SAMP}()$ |
| $T \leftarrow\!\!\$\ \mathbf{T}$ | $T \leftarrow\!\!\$\ \mathbf{T}$ |
| $(\alpha, x) \leftarrow F^T(\pi)$ | $(\alpha, x) \leftarrow F^T(\pi)$ |
| Return $(\alpha \geq 1)$ | $T' \leftarrow T$ |
| | $T'[x] \leftarrow\!\!\$\ \{0,1\}^\infty$ |
| | $(\alpha', x') \leftarrow F^{T'}(\pi)$ |
| | Return $((\alpha = \alpha') \wedge (\alpha \geq 1))$ |

**Fig. 1.** Games $\mathrm{G}^{\mathrm{single}}_{\mathrm{SAMP},F}$ (single run) and $\mathrm{G}^{\mathrm{double}}_{\mathrm{SAMP},F}$ (double run) associated with algorithms SAMP and $F$.

## 3 The Local Forking Lemma

We consider two algorithms SAMP and $F$. The first could be randomized but has no oracle. The second is deterministic and has access to a random oracle HASH as defined in Section 2. These algorithms work as follows.

Via $\pi \leftarrow\!\!\$\ \mathrm{SAMP}()$, algorithm SAMP returns a value $\pi$ that we think of as parameters that are input to $F$. Via $(\alpha, x) \leftarrow F^T(\pi)$, algorithm $F$, with input $\pi$, and with access to oracle $T \in \mathbf{T}$, returns a pair, where $\alpha \geq 0$ is an integer and $x$ is a string. We require that if $\alpha \geq 1$ then $x$ must be the $\alpha$-th query that $F$ has made to its oracle. If $\alpha = 0$, there is no requirement on $x$. Think of $\alpha = 0$ as denoting rejection and $\alpha \geq 1$ as denoting acceptance. We let $q$ denote maximum value that $\alpha$ can take. Furthermore, we require that the first $q$ queries that $F$ make must be distinct.

Consider the games $\mathrm{G}^{\mathrm{single}}_{\mathrm{SAMP},F}$ and $\mathrm{G}^{\mathrm{double}}_{\mathrm{SAMP},F}$ in Fig. 1. They are parameterized by algorithms SAMP and $F$. Game $\mathrm{G}^{\mathrm{single}}_{\mathrm{SAMP},F}$ is a "normal" execution, in which $\pi$ is sampled via SAMP, then $F$ is executed with oracle $T$, the game returned $\mathsf{true}$ if $\alpha \geq 1$ (acceptance) and $\mathsf{false}$ if $\alpha = 0$ (rejection). Game $\mathrm{G}^{\mathrm{double}}_{\mathrm{SAMP},F}$ begins with the same "normal" run. Then, it reruns $F$ with a different oracle $T'$. The difference is in just one point, namely the reply to the $\alpha$-th query. Otherwise, $T'$ is the same as $T$. This "local," as opposed to "global" change in $T'$ versus $T$ is the main difference from the General Forking Lemma of [7]. Our Local Forking Lemma relates the probability of these games returning $\mathsf{true}$. Our proof follows the template of [7].

**Lemma 1 (Local Forking Lemma).** *Let* SAMP, $F$ *and* $q$ *be as above. Then*

$$\Pr[\mathrm{G}^{\mathrm{double}}_{\mathrm{SAMP},F}] \geq \frac{1}{q} \cdot \Pr[\mathrm{G}^{\mathrm{single}}_{\mathrm{SAMP},F}]^2 \ . \tag{1}$$

*Proof (Lemma 1).* Consider the games of Figure 2. They are like the corresponding games of Figure 1 except that $\pi \in [\mathrm{SAMP}()]$ is fixed as a parameter of the game rather than chosen via SAMP in the game. Our main claim, that we will

| $\underline{\text{GAME } \mathrm{G}^{\text{single}}_{\pi,F}}$ | $\underline{\text{GAME } \mathrm{G}^{\text{double}}_{\pi,F}}$ |
|---|---|
| $T \leftarrow\!\!\$\ \mathbf{T}$ | $T \leftarrow\!\!\$\ \mathbf{T}$ |
| $(\alpha, x) \leftarrow F^T(\pi)$ | $(\alpha, x) \leftarrow F^T(\pi)$ |
| Return $(\alpha \geq 1)$ | $T' \leftarrow T$ |
| | $T'[x] \leftarrow\!\!\$\ \{0,1\}^\infty$ |
| | $(\alpha', x') \leftarrow F^{T'}(\pi)$ |
| | Return $((\alpha = \alpha') \wedge (\alpha \geq 1))$ |

**Fig. 2.** Games $\mathrm{G}^{\text{single}}_{\pi,F}$ (single run) and $\mathrm{G}^{\text{double}}_{\pi,F}$ (double run), with the parameter $\pi$ now fixed.

---

establish below, is that for every $\pi \in [\text{SAMP}()]$ we have

$$\Pr[\mathrm{G}^{\text{double}}_{\pi,F}] \geq \frac{1}{q} \cdot \Pr[\mathrm{G}^{\text{single}}_{\pi,F}]^2 \ . \tag{2}$$

From this we obtain Equation (1) as in [7]. Namely, define $\mathsf{Y}_1, \mathsf{Y}_2\colon\ [\text{SAMP}()] \to [0,1]$ by $\mathsf{Y}_1(\pi) = \Pr[\mathrm{G}^{\text{single}}_{\pi,F}]$ and $\mathsf{Y}_2(\pi) = \Pr[\mathrm{G}^{\text{double}}_{\pi,F}]$, and regard these as random variables over the choice of $\pi \leftarrow\!\!\$\ \text{SAMP}()$. Then, from Equation (2), we have

$$\begin{aligned}
\Pr[\mathrm{G}^{\text{double}}_{\text{SAMP},F}] &= \mathbf{E}[\mathsf{Y}_2] \\
&\geq \mathbf{E}\left[\frac{1}{q} \cdot \mathsf{Y}_1^2\right] \\
&\geq \frac{1}{q}\mathbf{E}\left[\mathsf{Y}_1\right]^2 \qquad (3) \\
&= \frac{1}{q} \cdot \Pr[\mathrm{G}^{\text{single}}_{\text{SAMP},F}]^2 \ ,
\end{aligned}$$

where Equation (3) is by Jensen's inequality. This establishes Equation (1). We proceed to the main task, namely to prove Equation (2). Henceforth, regard $\pi \in [\text{SAMP}()]$ as fixed.

Since $F$ makes a finite number of oracle queries and has finite running time, we can fix an integer $L$ such that any query $x$ made by $F$ has $|x| \leq L$ and also the maximum number of bits of any reply read by $F$ is at most $L$. This allows us to work over a finite sample space. Namely, let $D = \{0,1\}^{\leq L}$ be the set of all strings of length at most $L$ and let $R = \{0,1\}^L$ be shorthand for the set of strings of length $L$. Then let $\mathsf{OS}$ be the set of all functions $T\colon\ D \to R$. Now we can view $T$ in the games as being sampled from the finite set $\mathsf{OS}$.

We let $Q_1, Q_2, \ldots, Q_q$ denote the query functions of $F$, corresponding to the first $q$ queries. Function $Q_i\colon\ R^{i-1} \to D$ takes a list $h_1, \ldots, h_{i-1}$ of answers to queries $1, \ldots, i-1$ and returns the query that $F$ would make next. To be formal, the only possible input to $Q_1$ is the empty string $\varepsilon$, and it returns the first query made by $F$, which is uniquely defined since $F$ is deterministic. On input a string $h_1 \in R$, function $Q_2$ returns the query that $F$ would make if

it received $h_1$ as the answer to its first query. And so on, so that function $Q_i$, given $h_1, \ldots, h_{i-1} \in R$, returns the $i$-th query that $F$ would make had it received $h_1, \ldots, h_{i-1}$ as responses to its prior queries. We note again that the determinism of $F$ is important for these (deterministic) query functions to be well defined. For $i \in [1..q]$ we let $\mathbf{Q}(h_1, \ldots, h_{i-1}) = (Q_1(\varepsilon), Q_2(h_1), \ldots, Q_i(h_1, \ldots, h_{i-1}))$ be the vector consisting of the first $i$ queries given responses $h_1, \ldots, h_{i-1}$. Note that by our assumptions on $F$, the $i$ entries of this vector are always distinct.

We will be wanting to tinker with a function $T$, erasing it at some points, and then adding in new values. We now develop some language to facilitate this. If $V$ is a vector, we let $[V]$ denote the set whose elements are the entries of $V$, for example $[(1, 7, 5)] = \{1, 7, 5\}$. For a vector $Q \in D^i$ of possible queries, we let $\mathsf{OS}_Q$ denote the set of all functions $S\colon\ D \setminus [Q] \to R$, meaning functions just like those in $\mathsf{OS}$ but undefined at inputs in $[Q]$. Now if $S \in \mathsf{OS}_Q$ and $H \in R^i$ is a vector of possible answers, we let $S[H]$ denote the function $T \in \mathsf{OS}$ that reprograms $S$ on the query points, leaving it intact on others. In detail, for $1 \le j \le i$ we let $T(Q[j]) = H[j]$, and for $x \notin [Q]$, we let $T(x) = S(x)$.

Recall that $F$'s output is a pair of the form $(\alpha, x)$ where $0 \le \alpha \le q$ is an integer. We are only interested in the first output $\alpha$, and it is convenient to let $F_1$ denote the algorithm that returns this. Also if $i, \alpha \ge 0$ are integers, $\mathrm{Ind}_i(\alpha)$ is defined to be 1 if $\alpha = i$ and 0 otherwise. Now suppose $i \in [1..q]$. We let $\Omega_i$ be the set of all $(h_1, \ldots, h_{i-1}, S)$ such that $h_1, \ldots, h_{i-1} \in R$ and $S \in \mathsf{OS}_{\mathbf{Q}(h_1,\ldots,h_{i-1})}$, meaning $S$ is undefined at the first $i$ queries made by $F$. The function $\mathsf{X}_i\colon\ \Omega_i \to [0, 1]$ is then defined by

$$\begin{aligned}\mathsf{X}_i(h_1, \ldots, h_{i-1}, S) &= \Pr\left[\alpha = i \ :\ h \leftarrow\!\!\$\ R\,;\ \alpha \leftarrow F_1^{S[(h_1,\ldots,h_{i-1},h)]}(\pi)\right] \\ &= \frac{1}{|R|} \cdot \sum_{h \in R} \mathrm{Ind}_i\left(\ F_1^{S[(h_1,\ldots,h_{i-1},h)]}(\pi)\ \right)\ .\end{aligned}$$

This function fixes the answers to the first $i-1$ queries, which uniquely determines the $i$-th query, and also fixes, as $S$ the answers to all but these $i$ queries, taking the probability only over the answer $h$ to the $i$-th query. Let $\mathsf{I}$ and $\mathsf{I}'$ be the random variables taking values $\alpha$ and $\alpha'$, respectively, in game $\mathrm{G}^{\mathrm{double}}_{\pi,F}$. Then

$$\begin{aligned}\Pr[\mathrm{G}^{\mathrm{double}}_{\pi,F}] &= \Pr[\mathsf{I} \ge 1 \wedge \mathsf{I}' = \mathsf{I}] \\ &= \sum_{i=1}^{q} \Pr[\mathsf{I} = i \wedge \mathsf{I}' = i] \\ &= \sum_{i=1}^{q} \Pr[\,\mathsf{I} = i\,] \cdot \Pr[\,\mathsf{I}' = i \,|\, \mathsf{I} = i\,] \\ &= \sum_{i=1}^{q} \frac{1}{|\Omega_i|} \sum_{(h_1,\ldots,h_{i-1},S) \in \Omega_i} \mathsf{X}_i(h_1, \ldots, h_{i-1}, S)^2 \\ &= \sum_{i=1}^{q} \mathbf{E}[\mathsf{X}_i^2] \qquad (4)\end{aligned}$$

$$\geq \sum_{i=1}^{q} \mathbf{E}[\mathsf{X}_i]^2 \ . \tag{5}$$

In Equation (4), we regard $\mathsf{X}_i$ as a random variable over $\Omega_i$, and refer to its expectation. Equation (5) is by Jensen's inequality. Now recall that if $q \geq 1$ is an integer and $x_1, \ldots, x_q \geq 0$ are real numbers, then

$$q \cdot \sum_{i=1}^{q} x_i^2 \geq \left( \sum_{i=1}^{q} x_i \right)^2 \ .$$

This can be shown via Jensen's inequality or the Cauchy-Schwartz inequality, and a proof is in [7]. Setting $x_i = \mathbf{E}[\mathsf{X}_i]$, we have

$$q \cdot \sum_{i=1}^{q} \mathbf{E}[\mathsf{X}_i]^2 \geq \left( \sum_{i=1}^{q} \mathbf{E}[\mathsf{X}_i] \right)^2 \ .$$

At this point, we would like to invoke linearity of expectation to say that $\mathbf{E}[\mathsf{X}_1] + \cdots + \mathbf{E}[\mathsf{X}_q] = \mathbf{E}[\mathsf{X}_1 + \cdots + \mathsf{X}_q]$, but there is a difficulty, namely that linearity of expectation only makes sense when the random variables are over the same sample space, and ours are not, so the sum is not really even defined. (This is glossed over in [7].) So instead we expand the expectations again,

$$\begin{aligned} \sum_{i=1}^{q} \mathbf{E}[\mathsf{X}_i] &= \sum_{i=1}^{q} \frac{1}{|\Omega_i|} \sum_{(h_1,\ldots,h_{i-1},S)\in\Omega_i} \mathsf{X}_i(h_1, \ldots, h_{i-1}, S) \\ &= \sum_{i=1}^{q} \Pr[\mathsf{I} = i] \\ &= \Pr[\mathsf{I} \geq 1] = \Pr[\mathrm{G}^{\mathrm{single}}_{\pi,F}] \ . \end{aligned}$$

Putting all the above together, we have Equation (2). □

## 4 Public-Key-Dependent Message-Recovery security

We start by recalling definitions for public-key encryption schemes.

Public-key encryption. A public-key encryption (PKE) scheme $\mathsf{PKE}$ defines PT algorithms $\mathsf{PKE.Kg}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec}$, the last deterministic. Algorithm $\mathsf{PKE.Kg}$ takes as input $1^\lambda$ and outputs a public encryption key $ek \in \{0,1\}^{\mathsf{PKE.ekl}(\lambda)}$ and a secret decryption key $dk$, where $\mathsf{PKE.ekl}\colon\ \mathbb{N} \to \mathbb{N}$ is the public-key length of $\mathsf{PKE}$. Algorithm $\mathsf{PKE.Enc}$ takes as input $1^\lambda, ek$ and a message $m$ with $|m| \in \mathsf{PKE.IL}(\lambda)$ to return a ciphertext $c \in \{0,1\}^{\mathsf{PKE.cl}(\lambda,|m|)}$, where $\mathsf{PKE.IL}$ is the input-length function of $\mathsf{PKE}$, so that $\mathsf{PKE.IL}(\lambda) \subseteq \mathbb{N}$ is the set of allowed input (message) lengths, and $\mathsf{PKE.cl}\colon\ \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is the ciphertext length function of $\mathsf{PKE}$. Algorithm $\mathsf{PKE.Dec}$ takes $1^\lambda, dk, c$ and outputs $m \in \{0,1\}^* \cup \{\perp\}$. Correctness requires that $\mathsf{PKE.Dec}(1^\lambda, dk, c) = m$ for all $\lambda \in \mathbb{N}$, all $(ek, dk) \in [\mathsf{PKE.Kg}(1^\lambda)]$ all $m$ with $|m| \in \mathsf{PKE.IL}(\lambda)$ and all $c \in [\mathsf{PKE.Enc}(1^\lambda, ek, m)]$. Let $\mathsf{PKE.rl}\colon\ \mathbb{N} \to \mathbb{N}$ denote the randomness-length function of $\mathsf{PKE}$, meaning $\mathsf{PKE.Enc}(1^\lambda, \cdot, \cdot)$ draws its coins at random from the set $\{0,1\}^{\mathsf{PKE.rl}(\lambda)}$.

| $\underline{\text{GAME } \mathbf{G}^{\$\text{ind}}_{\mathsf{PKE},A}(\lambda)}$ | $\underline{\text{LR}(m_0, m_1)}$ |
|---|---|
| $(ek, dk) \leftarrow\!\!\$\ \mathsf{PKE.Kg}(1^\lambda)$ | If $(\lvert m_0\rvert \neq \lvert m_1\rvert)$ Return $\bot$ |
| $b \leftarrow\!\!\$\ \{0,1\}$ | $c \leftarrow\!\!\$\ \mathsf{PKE.Enc}(1^\lambda, ek, m_b)$ |
| $b' \leftarrow\!\!\$\ A^{\text{LR},\text{DEC}}(1^\lambda, ek)$ | $S \leftarrow S \cup \{c\}$ |
| Return $(b = b')$ | Return $c$ |
| $\underline{\text{HASH}(x, \ell)}$ | $\underline{\text{DEC}(c)}$ |
| If not $T[x,\ell]$ then | If $c \in S$ then return $\bot$ |
| $\quad T[x,\ell] \leftarrow\!\!\$\ \{0,1\}^\ell$ | $m \leftarrow\!\!\$\ \mathsf{PKE.Dec}(1^\lambda, dk, c)$ |
| Return $T[x,\ell]$ | Return $m$ |

**Fig. 3.** Game $\mathbf{G}^{\$\text{ind}}$ defining \$IND security of PKE.

Via game $\mathbf{G}^{\$\text{ind}}_{\mathsf{PKE},A}(\lambda)$ of Fig. 3, we recall the definition of what is usually called IND-CCA. We use the notation \$IND to emphasize that this is for randomized schemes and to avoid confusion with "IND" also being a notion for D-PKE schemes [5], and we cut the "CCA" for succinctness. We explicitly write the random oracle HASH as a variable-output-length one, so that it takes a string $x$ and integer $\ell$ to return a random $\ell$-bit string. (This can be implemented as discussed in Section 2 via a RO that, like in Lemma 1, takes one string input and returns strings of infinite length.) We let

$$\mathsf{Adv}^{\$\text{ind}}_{\mathsf{PKE},A}(\lambda) = 2\Pr[\mathbf{G}^{\$\text{ind}}_{\mathsf{PKE},A}(\lambda)] - 1\ .$$

We say that PKE is \$IND-secure if the function $\mathsf{Adv}^{\$\text{ind}}_{\mathsf{PKE},A}(\cdot)$ is negligible for every PT adversary $A$. We don't have to define what is conventionally called IND-CPA separately, but can recover it by saying that PKE is \$IND-CPA secure if the function $\mathsf{Adv}^{\$\text{ind}}_{\mathsf{PKE},A}(\cdot)$ is negligible for every PT adversary $A$ that makes zero queries to the DEC oracle.

We say that a PKE scheme PKE is a deterministic public-key encryption (D-PKE) [2] scheme if the encryption algorithm DE.Enc is deterministic. Formally, $\mathsf{PKE.rl}(\cdot) = 0$, so that the randomness can only be the empty string.

The EwH D-PKE scheme. We recall the Encrypt-with-Hash D-PKE scheme (formally, a transform) [2]. Let PKE be a PKE scheme. Then $\mathsf{DE} = \mathsf{EwH}[\mathsf{PKE}]$ is a ROM scheme defined as follows. First, $\mathsf{DE.Kg} = \mathsf{PKE.Kg}$ and $\mathsf{DE.Dec} = \mathsf{PKE.Dec}$, meaning the key generation and decryption algorithms of DE are the same as those of PKE. We also have that $\mathsf{DE.IL}(\lambda) = \mathsf{PKE.IL}(\lambda)$ and $\mathsf{DE.cl}(\lambda,\ell) = \mathsf{PKE.cl}(\lambda,\ell)$, for all $\lambda$ and message lengths $\ell$. We let $\mathsf{DE.rl}(\lambda) = 0$ for all $\lambda$. The encryption algorithm of DE is as follows:

$$\underline{\mathsf{DE.Enc}^{\text{HASH}}(1^\lambda, ek, m)}$$
$$r \leftarrow \text{HASH}(ek \| m, \mathsf{PKE.rl}(\lambda))\ ;\ c \leftarrow \mathsf{PKE.Enc}(1^\lambda, ek, m; r)$$
$$\text{Return } c$$

Above, HASH is the variable output length random oracle as discussed previously.

| $\underline{\text{GAME } \mathbf{G}^{\text{pred}}_{\mathsf{PKE},\mathcal{S},P}(\lambda)}$ | $\underline{\text{GAME } \mathbf{G}^{\text{pdmr}}_{\mathsf{PKE},S,A}(\lambda)}$ |
|---|---|
| $(ek, dk) \leftarrow\!\!\$\ \mathsf{PKE.Kg}(1^\lambda)$ | $(ek, dk) \leftarrow\!\!\$\ \mathsf{PKE.Kg}(1^\lambda)$ |
| $cc \leftarrow\!\!\$\ S.\mathsf{cx}(1^\lambda)$ | $cc \leftarrow\!\!\$\ S.\mathsf{cx}(1^\lambda)$ |
| $\mathbf{m} \leftarrow\!\!\$\ S.\mathsf{msg}^{\text{HASH,DEC}}(1^\lambda, cc, ek)$ | $\mathbf{m} \leftarrow\!\!\$\ S.\mathsf{msg}^{\text{HASH,DEC}}(1^\lambda, cc, ek)$ |
| For $i = 1, \ldots, \lvert\mathbf{m}\rvert$ do | For $i = 1$ to $\lvert\mathbf{m}\rvert$ do |
| $\quad \boldsymbol{\ell}[i] \leftarrow \lvert\mathbf{m}[i]\rvert$ | $\quad \mathbf{c}[i] \leftarrow\!\!\$\ \mathsf{PKE.Enc}^{\text{HASH}}(1^\lambda, ek, \mathbf{m}[i])$ |
| $(m, i) \leftarrow\!\!\$\ P^{\text{HASH,DEC}}(1^\lambda, cc, ek, \lvert\mathbf{m}\rvert, \boldsymbol{\ell})$ | $(m, i) \leftarrow\!\!\$\ A^{\text{HASH,DEC}}(1^\lambda, cc, ek, \mathbf{c})$ |
| Return $(m = \mathbf{m}[i])$ | Return $(m = \mathbf{m}[i])$ |
| $\underline{\text{HASH}(x, 1^\ell)}$ | $\underline{\text{HASH}(x, 1^\ell)}$ |
| If not $T[x, \ell]$ then | If not $T[x, \ell]$ then |
| $\quad T[x, \ell] \leftarrow\!\!\$\ \{0,1\}^\ell$ | $\quad T[x, \ell] \leftarrow\!\!\$\ \{0,1\}^\ell$ |
| Return $T[x, \ell]$ | Return $T[x, \ell]$ |
| $\underline{\text{DEC}(c)}$ | $\underline{\text{DEC}(c)}$ |
| Return $\mathsf{DE.Dec}(1^\lambda, dk, c)$ | If $(\exists i : c = \mathbf{c}[i])$ then Return $\perp$ |
| | Return $\mathsf{PKE.Dec}(1^\lambda, dk, c)$ |

**Fig. 4.** Left: Game defining unpredictability of source $S$. Right: Game defining PDMR security of PKE scheme $\mathsf{PKE}$ with source $S$ and PDMR adversary $A$.

---

PDMR. We know that D-PKE cannot provide indistinguishability-style security for messages that depend on the public key [2]. We ask whether, for public-key dependent messages, it could nonetheless provide a form of security that, although weaker, is desirable and meaningful in practice, namely security against message recovery. Here we give the necessary definitions, but in the general setting of PKE instead of restricting to D-PKE.

Let $\mathsf{PKE}$ be a PKE scheme. A source $S$ for $\mathsf{PKE}$ specifies PT algorithms $S.\mathsf{cx}$ and $S.\mathsf{msg}$, the first called the *context sampler* and the second called the *message sampler*. A PDMR adversary for source $S$ is an algorithm $A$. We associate to $\mathsf{PKE}$, $S$, $A$, and $\lambda \in \mathbb{N}$ the game $\mathbf{G}^{\text{pdmr}}_{\mathsf{PKE},S,A}(\lambda)$ in the right panel of Fig. 4. Via $cc \leftarrow\!\!\$\ S.\mathsf{cx}(1^\lambda)$, the game samples the *context*. Via $\mathbf{m} \leftarrow\!\!\$\ S.\mathsf{msg}^{\text{HASH,DEC}}(1^\lambda, cc, ek)$, the message sampler $S.\mathsf{msg}$ produces a target-message vector $\mathbf{m}$. We require that $\lvert\mathbf{m}[i]\rvert \in \mathsf{PKE.IL}(\lambda)$ for all $i$. *The fact that* $S.\mathsf{msg}$ *has* $ek$ *as input means that target messages may depend on the public key.* For $i = 1, \ldots, \lvert\mathbf{m}\rvert$, the game then encrypts message $\mathbf{m}[i]$ to create target ciphertext $\mathbf{c}[i]$. Via $(m, i) \leftarrow\!\!\$\ A^{\text{HASH,DEC}}(1^\lambda, cc, ek, \mathbf{c})$, the adversary $A$ produces a (guess) message $m$ and an index $i$ in the range $1 \le i \le \lvert\mathbf{c}\rvert$; it is guessing that $\mathbf{m}[i] = m$, and wins if this guess is correct. Note that $A$ is not allowed to query DEC on any of the ciphertexts in the vector $\mathbf{c}$. The PDMR-advantage $\mathsf{Adv}^{\mathsf{pdmr}}_{\mathsf{PKE},S,A}(\lambda) = \Pr[\mathbf{G}^{\text{pdmr}}_{\mathsf{PKE},S,A}(\lambda)]$ of $A$ is the probability that the game returns $\mathsf{true}$. For convenience of notation, we omit writing DEC in the superscript if the source or adversary do not query it.

Classes of sources. We define classes of sources (a set of message samplers) as a convenient way to state our results. For $n\colon\ \mathbb{N} \to \mathbb{N}$, we let $\mathcal{S}^n$ denote the class of

| | |
|---|---|
| $\mathcal{S}^n$ | Sources that output $n(\lambda)$ messages |
| $\mathcal{S}^{\mathsf{up}}$ | Unpredictable sources |
| $\mathcal{S}^{\mathsf{ri}}$ | Resampling-indistinguishable sources |

**Fig. 5.** Classes of message samplers of interest. See text for explanations.

sources whose message sampler's output vector $\mathbf{m} \leftarrow S.\mathsf{msg}^{\text{HASH,DEC}}(1^\lambda, \cdot, \cdot)$ has length $|\mathbf{m}| = n(\lambda)$. In some of our usage, $n$ will be a constant and we will refer, for example to $\mathcal{S}^1$ or $\mathcal{S}^2$. Later we will define other classes as well. A summary is in Figure 5.

Unpredictability. We cannot expect PDMR security for predictable target messages. Indeed, if, say, there are $s$ known choices for $\mathbf{m}[1]$ then $A$ can return one of them at random to get PDMR advantage $1/s$. Alternatively, $A$ could encrypt all $s$ candidates and return the one whose encryption equals $\mathbf{c}[1]$, getting an advantage of 1. We formalize unpredictability of a source $S$ via game $\mathbf{G}^{\text{pred}}_{\mathsf{PKE},S,P}$ specified in the left panel of Fig. 4, associated to D-PKE scheme $\mathsf{PKE}$, source $S$ and an adversary $P$ that we call a predictor. Source $S$ is run as in the message-recovery game. Next, instead of running $A$, predictor $P$ is run and it tries to predict (guess) some component of $\mathbf{m}$. Unlike $A$, predictor $P$ is not given $\mathbf{c}$. Instead it gets $|\mathbf{m}|$, the lengths of all component messages of this vector, and $1^\lambda, cc, ek$. Note that $P$ gets the decryption oracle DEC, with no restrictions on querying it. Predictor $P$ wins the game if $m = \mathbf{m}[i]$. For $\lambda \in \mathbb{N}$ we define the prediction advantage of $P$ to be

$$\mathsf{Adv}^{\mathsf{pred}}_{\mathsf{PKE},S,P}(\lambda) = \Pr[\mathbf{G}^{\text{pred}}_{\mathsf{PKE},S,P}(\lambda)] \ .$$

For $\lambda \in \mathbb{N}$ we also define

$$\mathsf{Adv}^{\mathsf{pred}}_{\mathsf{PKE},S}(\lambda) = \max_P \mathsf{Adv}^{\mathsf{pred}}_{\mathsf{PKE},S,P}(\lambda) \ .$$

where the maximum is over all predictors $P$, with no limit on their running time or the number of HASH queries. We say that $S$ is *unpredictable* if $\mathsf{Adv}^{\mathsf{pred}}_{\mathsf{PKE},S}(\cdot)$ is negligible. We let $\mathcal{S}^{\mathsf{up}}$ be the class of all unpredictable sources $S$.

Parameterized security. We will see that achievability of PDMR security depends very much on the class (set) of sources. Let $\mathcal{S}$ be a class of sources. We say that PKE scheme $\mathsf{PKE}$ is PDMR-secure against $\mathcal{S}$ if $\mathsf{Adv}^{\mathsf{pdmr}}_{\mathsf{PKE},S,A}(\cdot)$ is negligible for all $S \in \mathcal{S}$ and all PT $A$. We say that PKE scheme $\mathsf{PKE}$ is PDMR-CPA-secure against $\mathcal{S}$ if $\mathsf{Adv}^{\mathsf{pdmr}}_{\mathsf{PKE},S,A}(\lambda)$ is negligible for all $S \in \mathcal{S}$ that make no DEC queries and all PT $A$ that make no DEC queries.

$IND implies PDMR. We show that $IND-security implies PDMR security for *randomized* PKE schemes. It is important that this *does not* apply to D-PKE schemes as these cannot achieve $IND security. Let $\mathsf{PKE}$ be a PKE scheme, $S$ be a source for $\mathsf{PKE}$ and $A$ be a PDMR adversary for $S$. The following implies that if $\mathsf{PKE}$ is $IND secure, then it is PDMR-secure against $\mathcal{S}^n \cap \mathcal{S}^{\mathsf{up}}$ for any

polynomial $n$. Since the reduction preserves the number of decryption queries, the result holds in that case as well.

**Proposition 1.** *Let* $\mathsf{PKE}$ *be a PKE scheme, and* $n$ *a polynomial. Let* $S \in \mathcal{S}^n$ *be a source for* $\mathsf{PKE}$ *and let* $A$ *be a PDMR adversary. The proof gives \$IND adversary* $B$ *and predictor* $P$ *such that*

$$\mathsf{Adv}^{\$\mathsf{ind}}_{\mathsf{PKE},B}(\lambda) + \mathsf{Adv}^{\mathsf{pred}}_{\mathsf{PKE},S,P}(\lambda) \geq \mathsf{Adv}^{\mathsf{pdmr}}_{\mathsf{PKE},S,A}(\lambda) \ .$$

*Furthermore, the resources of adversary* $B$ *and predictor* $P$ *relate to those of* $S$ *and* $A$ *as follows:*

$$q_B^{\mathrm{LR}} = n, \quad q_B^{\mathrm{Hash}} = q_S^{\mathrm{Hash}} + q_A^{\mathrm{Hash}}, \quad q_B^{\mathrm{Dec}} = q_S^{\mathrm{Dec}} + q_A^{\mathrm{Dec}}, \quad t_B \approx t_S + t_A \ ,$$

*and*

$$q_P^{\mathrm{Hash}} = q_A^{\mathrm{Hash}} + n \cdot q^{\mathrm{Hash}}_{\mathsf{PKE.Enc}}, \quad q_P^{\mathrm{Dec}} = q_A^{\mathrm{Dec}}, \quad t_P \approx n \cdot t_{\mathsf{PKE.Enc}} + t_A \ .$$

*Proof (of Proposition 1).* \$IND adversary $B$ and predictor $P$ are as follows:

| Adversary $B^{\mathrm{LR},\mathrm{Hash},\mathrm{Dec}}(1^\lambda, ek)$ | Adversary $P^{\mathrm{Hash},\mathrm{Dec}}(1^\lambda, cc, ek, \boldsymbol{\ell})$ |
|---|---|
| $cc \leftarrow\!\!\$\ S.\mathsf{cx}(1^\lambda)$ | For $i = 1, \ldots, \lvert\boldsymbol{\ell}\rvert$ do |
| $\mathbf{m} \leftarrow\!\!\$\ S.\mathsf{msg}^{\mathrm{Hash},\mathrm{Dec}}(1^\lambda, cc, ek)$ | $\quad \mathbf{m}'[i] \leftarrow\!\!\$\ \{0,1\}^{\boldsymbol{\ell}[i]}$ |
| For $i = 1, \ldots \lvert\mathbf{m}\rvert$ do | $\quad \mathbf{c}[i] \leftarrow\!\!\$\ \mathsf{PKE.Enc}^{\mathrm{Hash}}(1^\lambda, ek, \mathbf{m}'[i])$ |
| $\quad \mathbf{m}'[i] \leftarrow\!\!\$\ \{0,1\}^{\lvert\mathbf{m}[i]\rvert}$ | $(\overline{m}, i) \leftarrow\!\!\$\ A^{\mathrm{Hash},\mathrm{DecSim}}(1^\lambda, cc, ek, \mathbf{c})$ |
| $\quad \mathbf{c}[i] \leftarrow\!\!\$\ \mathrm{LR}(\mathbf{m}'[i], \mathbf{m}[i])$ | Return $(\overline{m}, i)$ |
| $(\overline{m}, i) \leftarrow\!\!\$\ A^{\mathrm{Hash},\mathrm{Dec}}(1^\lambda, cc, ek, \mathbf{c})$ | Algorithm $\mathrm{DecSim}(x)$ |
| Return $(\overline{m} = \mathbf{m}[i])$ | If $(\exists i : x = \mathbf{c}[i])$ then return $\perp$ |
| | Return $\mathrm{Dec}(x)$ |

Adversary $B$ uses $\mathbf{m}$ output by $S.\mathsf{msg}$ as well as $\mathbf{m}'$ that is sampled uniformly at random at each component $i$ subjected to $|\mathbf{m}[i]| = |\mathbf{m}'[i]|$. Adversary $B$ will query $\mathrm{LR}(m', m)$ to obtain ciphertext $c$. Adversary $B$ then runs $A$ on ciphertext $\mathbf{c}$ and checks if the guess of $A$ matches message $m$. Predictor $P$ obtains the encryption of a randomly sampled messages $\mathbf{m}'$ where component $i$ has length $\boldsymbol{\ell}[i]$. Then it runs $A$ and returns its output. We have

$$\begin{aligned}
\mathsf{Adv}^{\$\mathsf{ind}}_{\mathsf{PKE},B}(\lambda) &= 2 \cdot \Pr[b = b'] - 1 \\
&= \Pr[b' = 1 \mid b = 1] - (1 - \Pr[b' = 0 \mid b = 0]) \\
&= \Pr[b' = 1 \mid b = 1] - \Pr[b' = 1 \mid b = 0] \ ,
\end{aligned}$$

where $b'$ and $b$ are random variables associated to game $\mathbf{G}^{\$\mathrm{ind}}_{\mathsf{PKE},B}(\lambda)$. It is standard to check that

$$\Pr[b' = 1 \mid b = 1] = \mathsf{Adv}^{\mathsf{pdmr}}_{\mathsf{PKE},S,A}(\lambda) \ , \tag{6}$$

and

$$\Pr[b' = 1 \mid b = 0] = \mathsf{Adv}^{\mathsf{pred}}_{\mathsf{PKE},S,P}(\lambda) \ . \tag{7}$$

Combining the above two equations, we obtain Proposition 1. □

## 5 Possibility Results

In this section, we show that when messages are not too strongly related to each other —more precisely when they are resampling-indistinguishable, to be defined shortly— PDMR security is possible. Furthermore this is not just in principle, but in practice: we show that such PDMR security is provided by the simple and efficient EwH scheme. Thus we can add, to the IND security for public-key independent messages we know this scheme already provides [2], a good privacy guarantee for messages that depend on the public key. This supports the security of existing or future uses of the scheme.

In more detail, our main technical result, Theorem 1, shows that $\mathsf{DE} = \mathsf{EwH}[\mathsf{PKE}]$ is PDMR-secure against $\mathcal{S}^1$ sources (namely, for the encryption of a single message) as long as the same is true for the randomized PKE. The proof relies crucially on Lemma 1. Note that this reduction does not need to assume unpredictability of the source. It follows from Proposition 1 that $\mathsf{DE} = \mathsf{EwH}[\mathsf{PKE}]$ is PDMR-secure against $\mathcal{S}^{\mathsf{up}} \cap \mathcal{S}^1$ sources as long as the randomized PKE is \$IND-secure.

The above is all for encryption of a single message. We will then turn to the encryption of multiple messages. We define a class of sources $\mathcal{S}^{\mathsf{ri}}$ that we call resampling indistinguishable. Such sources produce a polynomially-long vector of messages, reflecting that we are asking for privacy when encrypting many messages. Theorem 2 is a general result saying that *any* scheme that is PDMR-secure for $\mathcal{S}^1 \cap \mathcal{S}^{\mathsf{up}}$ is automatically PDMR security for $\mathcal{S}^{\mathsf{ri}} \cap \mathcal{S}^{\mathsf{up}}$, meaning PDMR-security for a single unpredictable message implies it for any polynomial number of unpredictable resampling-indistinguishable messages. Putting all this together, we get that $\mathsf{DE} = \mathsf{EwH}[\mathsf{PKE}]$ is PDMR-secure against $\mathcal{S}^{\mathsf{up}} \cap \mathcal{S}^{\mathsf{ri}}$ sources as long as the randomized PKE is \$IND-secure.

Remark regarding CPA. All of the results in this section are stated in the presence of a decryption oracle. However, our reductions will preserve the number of decryption queries, so that analogous CPA-type result can be obtained simply by restricting the number of decryption queries $q^{\mathrm{Dec}}$ to be 0 for all sources and adversaries involved. Thus the statement "... PDMR(-CPA)-secure ... \$IND(-CPA)-secure ...", is read as two separate statements: "... PDMR-secure ... \$IND-secure" and "... PDMR-CPA-secure ... \$IND-CPA-secure ...".

Remark regarding PKE schemes that rely on a random oracle. For simplicity we assume that the starting randomized PKE scheme is not a ROM scheme. However our result applies also to the case where it is in fact a ROM scheme like those of [9, 18]. For this, we simply use domain separation, effectively making the RO used by EwH and the RO used by PKE independent.

### 5.1 Security of EwH for a single message

Canonical 1-sources and PDMR adversaries. Let $S \in \mathcal{S}^1$ be a source for DE, and $A$ be a PDMR adversary for $S$. Since $S.\mathsf{msg}$ only produces one message, we can

$B^{\text{HASH},\text{DEC}}(1^\lambda, cc, ek, c)$
$Q \leftarrow \emptyset$ ; $\mu \leftarrow\!\!\$\ A^{\text{HSIM},\text{DEC}}(1^\lambda, cc, ek, c)$
$x \leftarrow\!\!\$\ Q$ ; $(ek \| m^*, \ell) \leftarrow x$
Return $m^*$

$\text{HSIM}(w)$
$Q \leftarrow Q \cup \{w\}$
Return $\text{HASH}(w)$

Algorithm SAMP
$(ek, dk) \leftarrow\!\!\$\ \mathsf{PKE.Kg}(1^\lambda)$
$cc \leftarrow\!\!\$\ S.\mathsf{cx}(1^\lambda)$
$\rho_S \leftarrow\!\!\$\ \{0,1\}^{S.\mathsf{msg.rl}(\lambda)}$
$\rho_A \leftarrow\!\!\$\ \{0,1\}^{A.\mathsf{rl}(\lambda)}$
Return $(ek, dk, cc, \rho_S, \rho_A)$

Subroutine DECSIM($d$)
If $(d = c)$ then return $\perp$
Return $\mathsf{PKE.Dec}(1^\lambda, dk, d)$

Algorithm $F^{\text{HASH}}((ek, dk, cc, \rho_S, \rho_A))$
$j \leftarrow 0$ ; $Q \leftarrow \emptyset$ ; $\mathsf{ms} \leftarrow \mathsf{true}$
$m \leftarrow S.\mathsf{msg}^{\text{HSIM},\text{DECSIM}}(1^\lambda, cc, ek; \rho_S)$
$x \leftarrow (ek \| m, \mathsf{PKE.rl})$ ; $r \leftarrow \text{HASH}(x)[1..\mathsf{PKE.rl}]$
$c \leftarrow \mathsf{PKE.Enc}(1^\lambda, ek, m; r)$ ; $\mathsf{ms} \leftarrow \mathsf{false}$
$\mu \leftarrow A^{\text{HASH},\text{DECSIM}}(1^\lambda, cc, ek, c; \rho_A)$
If $(x \in Q)$ then $\alpha \leftarrow \text{Idx}(x)$ else $\alpha \leftarrow 0$
Return $(\alpha, x)$

Subroutine HSIM($w$)
If $(\mathsf{ms})$ then $j \leftarrow j + 1$ ; $\text{Idx}(w) \leftarrow j$
Else $Q \leftarrow Q \cup \{w\}$
Return $\text{HASH}(w)$

**Fig. 6.** Top is our PDMR adversary $B$ against PKE in the proof of Theorem 1. It invokes a given PDMR adversary $A$ against EwH[PKE]. Bottom are algorithms SAMP, $F$ used in the analysis.

assume that the message index given by $A$ is always 1. Hence, we can view the output of both $S.\mathsf{msg}$ and $A$ as a single message. Next, we note that we can require $S.\mathsf{msg}$ and $A$ to query HASH at $(ek\|m, \mathsf{PKE.rl}(\lambda))$ if they output $m$. This can always be done at the expense of one more query to HASH. For the following results, we shall assume canonical 1-sources and PDMR adversaries for them.

PDMR-security of PKE implies PDMR of EwH. The following says that if randomized scheme PKE is PDMR-secure for a source $S \in \mathcal{S}^1$, then so is deterministic scheme $\mathsf{DE} = \mathsf{EwH}[\mathsf{PKE}]$. As noted above, the theorem itself does not assume unpredictability of the source. That will enter later.

**Theorem 1.** *Let* PKE *be a public-key encryption scheme. Let* $\mathsf{DE} = \mathsf{EwH}[\mathsf{PKE}]$ *be the associated deterministic public-key encryption scheme. Let* $S \in \mathcal{S}^1$ *be a 1-message source. Let* $A$ *be a PDMR adversary for* $S$*, and let* $B$ *be the PDMR adversary for* $S$ *given in Fig. 6. Then*

$$\mathsf{Adv}^{\mathsf{pdmr}}_{\mathsf{PKE},S,B}(\lambda) \geq \frac{1}{(1 + q_S^{\text{HASH}}) \cdot (1 + q_A^{\text{HASH}})} \cdot \left(\mathsf{Adv}^{\mathsf{pdmr}}_{\mathsf{DE},S,A}(\lambda)\right)^2 .$$

*Additionally* $q_B^{\text{HASH}} \leq 1 + q_A^{\text{HASH}}$, $q_B^{\text{DEC}} = q_A^{\text{DEC}}$ *and* $t_B \approx t_A + \mathcal{O}(q_A^{\text{HASH}})$.

*Proof (of Theorem 1).* Let $\ell = \mathsf{PKE.rl}$. We assume that if $S^{\text{HASH},\text{DEC}}(1^\lambda, cc, ek)$ outputs message $m$ then it has always queried $(ek\|m, \ell)$ to HASH. Likewise, we assume that if $A^{\text{HASH},\text{DEC}}(1^\lambda, cc, ek, c)$ outputs message $\mu$ then it has always

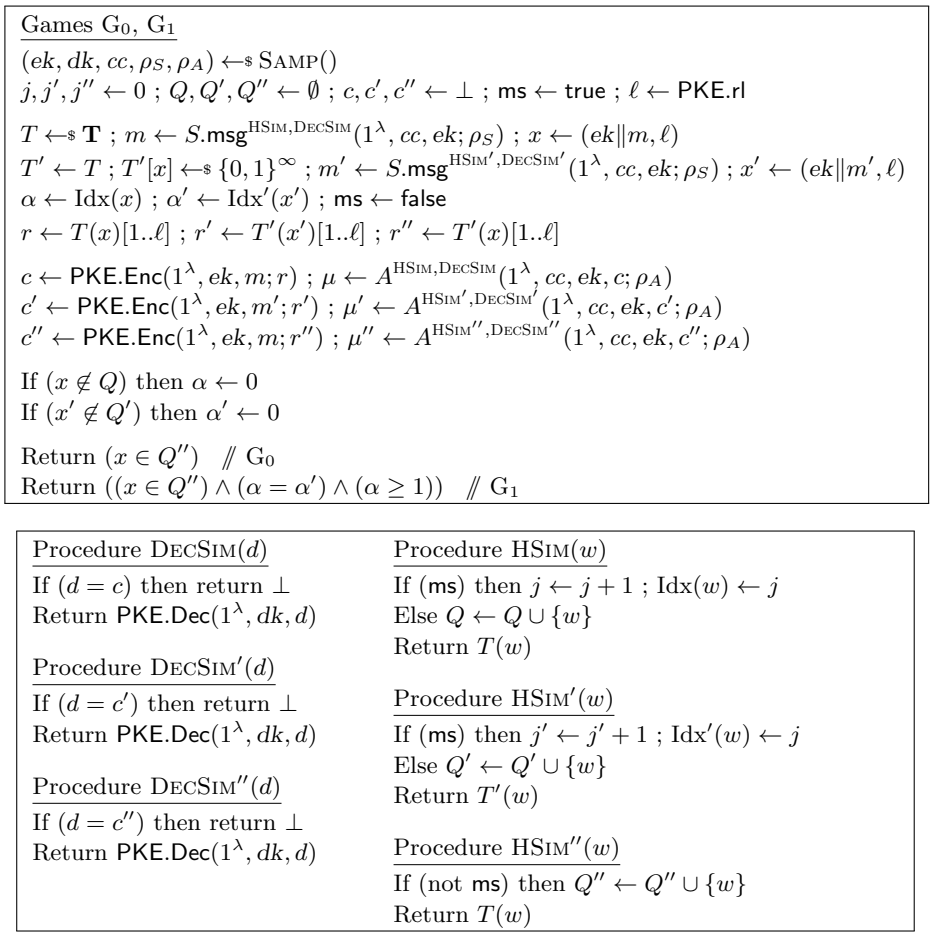


**Fig. 7.** Games $G_0, G_1$ for proof of Theorem 1, in the top box, differ only in their Return statements, and use the procedures in the bottom box.

queried $(ek\|\mu, \ell)$ to Hash. In both cases, as discussed above, this can be ensured by modifying the algorithm to make the required query if it did not already do so, increasing the number of Hash queries by at most one. So, letting $q_1 = q_S^{\textsc{Hash}}$ and $q_2 = q_A^{\textsc{Hash}}$, we now regard the number of Hash queries of $S$ and $A$ as $1 + q_1$ and $1 + q_2$, respectively. We assume that all Hash queries of $S$ are distinct, and also that all Hash queries of $A$ are distinct. Crucially, we do not, and cannot, assume distinctness across these queries, meaning $A$ could repeat queries made by $S$.

Fix some $\lambda \in \mathbb{N}$. We start the analysis with the Samp algorithm of Figure 6. (Ignore the rest of that Figure for now.) It picks keys, common coins $cc$, coins $\rho_S$ for the message-finding phase of sampler $S$, and coins $\rho_A$ for $A$, so that these

Games $\mathrm{G}_2$, $\mathrm{G}_3$

$(ek, dk, cc, \rho_S, \rho_A) \leftarrow\!\!\$\ \textsc{Samp}()$
$j, j', j'' \leftarrow 0\ ;\ Q, Q', Q'' \leftarrow \emptyset\ ;\ c, c', c'' \leftarrow \bot\ ;\ \mathsf{ms} \leftarrow \mathsf{true}\ ;\ \ell \leftarrow \mathsf{PKE.rl}$

$T \leftarrow\!\!\$\ \mathbf{T}\ ;\ m \leftarrow S.\mathsf{msg}^{\textsc{HSim},\textsc{DecSim}}(1^\lambda, cc, ek; \rho_S)\ ;\ x \leftarrow (ek \| m, \ell)$
$T' \leftarrow T\ ;\ T'[x] \leftarrow\!\!\$\ \{0,1\}^\infty\ ;\ m' \leftarrow S.\mathsf{msg}^{\textsc{HSim}',\textsc{DecSim}'}(1^\lambda, cc, ek; \rho_S)\ ;\ x' \leftarrow (ek \| m', \ell)$
$\alpha \leftarrow \mathrm{Idx}(x)\ ;\ \alpha' \leftarrow \mathrm{Idx}'(x')\ ;\ \mathsf{ms} \leftarrow \mathsf{false}$
$r \leftarrow T(x)[1..\ell]\ ;\ r' \leftarrow T'(x)[1..\ell]$

$c \leftarrow \mathsf{PKE.Enc}(1^\lambda, ek, m; r)\ ;\ \mu \leftarrow A^{\textsc{HSim},\textsc{DecSim}}(1^\lambda, cc, ek, c; \rho_A)$
$c' \leftarrow \mathsf{PKE.Enc}(1^\lambda, ek, m; r')\ ;\ \mu' \leftarrow A^{\textsc{HSim}',\textsc{DecSim}'}(1^\lambda, cc, ek, c'; \rho_A)$

$\mu'' \leftarrow A^{\textsc{HSim}'',\textsc{DecSim}'}(1^\lambda, cc, ek, c'; \rho_A)$ // $\mathrm{G}_2$
$\mu'' \leftarrow \mu'$ // $\mathrm{G}_3$

If $(x \notin Q)$ then $\alpha \leftarrow 0$
If $(x \notin Q')$ then $\alpha' \leftarrow 0$

Return $((x \in Q'') \wedge (\alpha = \alpha') \wedge (\alpha \geq 1))$ // $\mathrm{G}_2$
Return $((x \in Q') \wedge (\alpha = \alpha') \wedge (\alpha \geq 1))$ // $\mathrm{G}_3$

**Fig. 8.** Games $\mathrm{G}_2, \mathrm{G}_3$ for the proof of Theorem 1. They use the procedures at the bottom of Figure 7.

can be fixed and maintained across multiple executions of the algorithms. Now consider games $\mathrm{G}_0, \mathrm{G}_1$ at the top of Figure 7. They invoke Samp at the very beginning. They also invoke the procedures in the bottom of Figure 7. We claim that

$$\mathsf{Adv}^{\mathsf{pdmr}}_{\mathsf{PKE},S,B}(\lambda) \geq \frac{1}{1+q_2} \cdot \Pr[\mathrm{G}_0]\ . \tag{8}$$

This is justified as follows. The message $m$ in $\mathrm{G}_0$ is created just as in game $\mathbf{G}^{\mathrm{pdmr}}_{\mathsf{PKE},S,A}(\lambda)$, the oracle Hash being set, by procedure HSim, to $T$. In game $\mathbf{G}^{\mathrm{pdmr}}_{\mathsf{PKE},S,A}(\lambda)$, ciphertext $c$ is created by encryption of $m$ under coins that are random and independent of Hash, captured in $\mathrm{G}_0$ as $T'[x]$. However, $B$ runs $A$ with its own oracle Hash, here $T$, not $T'$, captured in $\mathrm{G}_0$ as $\textsc{HSim}''$. We have written HSim and $\textsc{HSim}''$ as two, separate, oracles, even though both reply simply via $T$, because they keep track of different things. In the message-sampling phase (flag $\mathsf{ms} = \mathsf{true}$) they store the index of each query, and when $A$ is run (flag $\mathsf{ms} = \mathsf{false}$), they store the queries in a set. Note that in $\mathrm{G}_0$, we are not concerned with $x', r', c', \mu, \mu', \alpha, \alpha'$, meaning all these quantities can be ignored in the context of Equation (8). Game $\mathrm{G}_0$ returns $\mathsf{true}$ if $x \in Q''$, meaning if $A$ made query $x = (ek \| m, \ell)$ to $\textsc{HSim}''$. We have assumed that $A$ always makes hash query $(ek \| \mu, \ell)$ on output $\mu$, and we have $|Q''| \leq 1 + q_2$, yielding Equation (8).

Games $\mathrm{G}_0, \mathrm{G}_1$ differ only in what they return, and the boolean returned by $\mathrm{G}_1$ is the one returned by $\mathrm{G}_0$ ANDed with more stuff. So, regardless of what is this

stuff, we must have

$$\Pr[\mathrm{G}_0] \geq \Pr[\mathrm{G}_1] \ . \tag{9}$$

Suppose the winning condition of game $\mathrm{G}_1$ is met, so that $\alpha = \alpha' \neq 0$. This implies $(x, m, r', c') = (x', m', r'', c'')$. To explain, we have assumed the hash queries of $S$ are distinct, we have maintained the coins of $S$ across the runs, and $T, T'$ differ only at $x$, so until $x$ is queried, the executions of $S$ are the same, so $x = x'$. This implies $r' = r''$. From the definitions of $x, x'$ we get $m = m'$, and thus we also get $c' = c''$. In game $\mathrm{G}_2$ of Figure 8 —the procedures used continue to be those at the bottom of Figure 7— we rewrite and simplify the code of $\mathrm{G}_1$ under the assumption that $(x, m, r', c') = (x', m', r'', c'')$. Since $\mathrm{G}_2$ maintains the winning condition of $\mathrm{G}_1$, and we have seen this implies $(x, m, r', c') = (x', m', r'', c'')$, we have

$$\Pr[\mathrm{G}_1] = \Pr[\mathrm{G}_2] \ . \tag{10}$$

In game $\mathrm{G}_2$, consider the computations of $\mu'$ and $\mu''$. The only difference is that in the first $A$ has oracle $\textsc{Hsim}'$, and in the second, $\textsc{Hsim}''$. However, the replies from these oracles differ only at query $x$, and the winning condition of $\mathrm{G}_2$ depends only on $x$ and other quantities determined prior to the reply to hash query $x$ being obtained by $A$. This means that the winning condition of game $\mathrm{G}_3$ is equivalent to that of $\mathrm{G}_2$. (Game $\mathrm{G}_3$ no longer computes $\mu''$ as in game $\mathrm{G}_2$ to ensure $\textsc{Hsim}''$ is no longer used, and sets $\mu''$ instead, correctly, to $\mu'$, but this quantity is not used in the winning condition.) We have

$$\Pr[\mathrm{G}_2] = \Pr[\mathrm{G}_3] \ . \tag{11}$$

Now consider algorithm $F$ of Figure 6, and consider executing game $\mathrm{G}^{\mathrm{double}}_{\mathrm{Samp},F}$ of Figure 1. We have

$$\Pr[\mathrm{G}_3] \geq \Pr[\mathrm{G}^{\mathrm{double}}_{\mathrm{Samp},F}] \tag{12}$$

$$\geq \frac{1}{1+q_2} \cdot \Pr[\mathrm{G}^{\mathrm{single}}_{\mathrm{Samp},F}]^2 \ , \tag{13}$$

where Equation 13 is by Lemma 1. Now we observe that

$$\Pr[\mathrm{G}^{\mathrm{single}}_{\mathrm{Samp},F}] \geq \mathsf{Adv}^{\mathsf{pdmr}}_{\mathsf{DE},S,A}(\lambda) \ . \tag{14}$$

Combining the equations above completes the proof. □

PDMR Security of EwH for unpredictable one-message sources. An immediate corollary of Proposition 1 and Theorem 1 is that \$IND(-CPA) security of PKE implies PDMR(-CPA)-security of EwH[PKE] against $\mathcal{S}^1 \cap \mathcal{S}^{\mathsf{up}}$.

**Corollary 1.** *Let* PKE *be a public-key encryption scheme. Let* $\mathsf{DE} = \mathsf{EwH}[\mathsf{PKE}]$ *be the associated deterministic public-key encryption scheme. Let* $S \in \mathcal{S}^1$ *be a 1-message source. Let* $A$ *be a PDMR adversary for* $S$*. The proof specifies PDMR adversary* $B$ *for* $S$*, and predictor* $P$*, such that*

$$\mathsf{Adv}^{\mathsf{pdmr}}_{\mathsf{DE},S,A}(\lambda) \leq \sqrt{(1+q_S^{\mathrm{Hash}})(1+q_A^{\mathrm{Hash}})\left(\mathsf{Adv}^{\mathsf{\$ind}}_{\mathsf{PKE},B}(\lambda) + \mathsf{Adv}^{\mathsf{pred}}_{\mathsf{PKE},S,P}(\lambda)\right)} \ .$$

| $\underline{\text{GAME } \mathbf{G}^{\text{ri}}_{\mathsf{DE},S,D}(\lambda)}$ | $\underline{\text{HASH}(x, 1^\ell)}$ |
|---|---|
| $(ek, dk) \leftarrow \mathsf{DE.Kg}(1^\lambda)$ | If not $T[x,\ell]$ then |
| $cc \leftarrow\!\!\$\ S.\mathsf{cx}(1^\lambda)$ ; $b \leftarrow\!\!\$\ \{0,1\}$ | $\quad T[x,\ell] \leftarrow\!\!\$\ \{0,1\}^\ell$ |
| $j \leftarrow\!\!\$\ [n(\lambda)]$ | Return $T[x,\ell]$ |
| $\mathbf{m}_0 \leftarrow\!\!\$\ S.\mathsf{msg}^{\text{HASH,DEC}}(1^\lambda, cc, ek)$ | $\underline{\text{DEC}(c)}$ |
| $\mathbf{m}_1 \leftarrow \mathbf{m}_0$ | Return $\mathsf{DE.Dec}(1^\lambda, dk, c)$ |
| $\mathbf{m}_1[j] \leftarrow\!\!\$\ S.\mathsf{msg}^{\text{HASH,DEC}}(1^\lambda, cc, ek)[j]$ | |
| $b' \leftarrow\!\!\$\ D^{\text{HASH,DEC}}(1^\lambda, cc, ek, \mathbf{m}_b, j)$ | |
| Return $(b = b')$ | |

**Fig. 9.** Game defining resampling indistinguishability of source $S$ for $\mathsf{DE}$.

*The resources of $B$ and $P$ are related to those of $S$ and $A$ as follows:*

$$q_B^{\text{HASH}} = q_S^{\text{HASH}} + q_A^{\text{HASH}}, \quad q_B^{\text{DEC}} = q_S^{\text{DEC}} + q_A^{\text{DEC}}, \quad t_B \approx t_S + t_A \ ,$$

*and*

$$q_P^{\text{HASH}} = q_A^{\text{HASH}} + q_{\mathsf{Enc}}^{\text{HASH}}, \quad q_P^{\text{DEC}} = q_A^{\text{DEC}}, \quad t_P \approx t_{\mathsf{Enc}} + t_A \ .$$

### 5.2 Resampling Indistinguishability

We define what it means for an adversary $A$ to be resampling indistinguishable. At a high level, the condition is that, the distribution of the vector of messages produced by the adversary is not detectably changed by replacing one of the components of the vector with a component from another vector produced by a second run of the adversary using independent coins. This captures a weak form of independence of the components of the vector. We give accompanying examples after the precise definition.

Definition. Let $\mathsf{DE}$ be a D-PKE scheme. Consider the game $\mathbf{G}^{\text{ri}}_{\mathsf{DE},S,D}$ given in Fig. 9, where $S$ is a $n(\lambda)$-source for $\mathsf{DE}$ and $D$ is an adversary called the resampling distinguisher. In this game, a message vector $\mathbf{m}_0$ is obtained by running $S.\mathsf{msg}$. Then $\mathbf{m}_1$ is created to be the same as $\mathbf{m}_0$ except at one, random, location $j$. The value it takes at $j$ is the $j$-th component of a message vector obtained by running $S.\mathsf{msg}$ again, independently and with fresh coins, but on the same inputs $1^\lambda, cc, ek$. Finally, $D$ takes input $(1^\lambda, cc, ek, \mathbf{m}_b, j)$ and attempts to guess the value of $b$. We let

$$\mathsf{Adv}^{\text{ri}}_{\mathsf{DE},S,D}(\lambda) = 2\Pr[\mathbf{G}^{\text{ri}}_{\mathsf{DE},S,D}(\lambda)] - 1 \ .$$

We say that $S$ is resampling-indistinguishable if the function $\mathsf{Adv}^{\text{ri}}_{\mathsf{DE},S,D}(\cdot)$ is negligible for any PT distinguisher $D$. We let $\mathcal{S}^{\text{ri}}$ be the class of resampling-indistinguishable sources.

Examples of message samples in $\mathcal{S}^{\text{ri}}$. We give some examples of RI sources. First, if each $\mathbf{m}[i]$ is sampled independently from some distribution depending on $i$,

then $S$ is RI *even when these distribution depends on the public key.* More precisely, suppose, for some PT algorithm $X$ and polynomial $n(\cdot)$, sampler $S.\mathsf{msg}$ works as follows:

$$\begin{array}{l}\underline{\text{Adversary } S.\mathsf{msg}^{\text{HASH}}(1^\lambda, cc, ek)}\\ \text{For } i = 1, \ldots, n(\lambda) \text{ do } \mathbf{m}[i] \leftarrow\!\!\$\; X^{\text{HASH}}(1^\lambda, cc, ek, i)\\ \text{Return } \mathbf{m}\end{array}$$

Then, for any choices of $X, n$, sampler $S.\mathsf{msg}$ as above (together with any context sampler) is RI. Moreover, $S$ is perfectly RI, i.e. $\mathsf{Adv}^{\mathsf{ri}}_{A,D}(\lambda) = 0$ for any distinguisher $D$. Note that the class of such adversaries, defined by all the choices of PT $X$ and polynomials $n$, is too large for the constructions of RSV [30], so our positive results give schemes providing security for classes of message distributions for which their schemes do not provide security. This example extends naturally to sources $S'$ such that the output of $S'.\mathsf{msg}$ is *indistinguishable* from the output of $S.\mathsf{msg}$ (for some choice of $X$ and $n(\cdot)$). The notion of RI also allows us to capture correlation in $\mathbf{m}$ that cannot be efficiently detected. For example, consider $S$ that does the following. It first generate a random string $r \leftarrow\!\!\$\; \{0,1\}^n$. Then, it sets $\mathbf{m}[i] \leftarrow \text{HASH}(r \| i, 1^n)$ for $i \in \{1, 2\}$. Note that there is strong information-theoretic correlation between $\mathbf{m}[1]$ and $\mathbf{m}[2]$, given the entire function table of HASH. However, any distinguisher $D$ making $q$ queries to HASH cannot detect this correlation with advantage more than $q/2^n$. Finally, we note that resampling-indistinguishability is independent of predictability. In particular, if $X$ always returns a constant message (that is compatible with the message space of the encryption scheme), then the source constructed before is still RI, but it is trivially predictable.

Reduction to 1-PDMR security. A useful property of RI adversaries is that their PDMR security reduces to the PDMR security of the encryption of just one message. This is formalized via the theorem below, which says that $\mathsf{DE}$ is PDMR-(CCA-)secure for $\mathcal{S}^{\mathsf{up}} \cap \mathcal{S}^{1}$, then it is PDMR-(CCA-)secure for $\mathcal{S}^{\mathsf{up}} \cap \mathcal{S}^{\mathsf{ri}}$.

**Theorem 2.** *Let* $\mathsf{DE}$ *be any D-PKE scheme. Let* $S_1$ *be any* $n(\lambda)$*-source and* $A$ *be a PDMR adversary for D-PKE scheme* $\mathsf{DE}$*. Consider the* 1*-source* $S_2$ *and PDMR adversary* $B$ *given in Fig. 10. Then*

$$\mathsf{Adv}^{\mathsf{pdmr}}_{\mathsf{DE},S_1,A}(\lambda) \le n(\lambda) \cdot \left(\mathsf{Adv}^{\mathsf{pdmr}}_{\mathsf{DE},S_2,B}(\lambda) + \mathsf{Adv}^{\mathsf{ri}}_{\mathsf{DE},S_1,D}(\lambda)\right) . \tag{15}$$

*Source* $S_2$*, adversary* $B$*, and distinguisher* $D$ *are efficient as long as* $S_1$ *and* $A$ *are. In particular,*

$$q_B^{\text{HASH}} = q_{S_1}^{\text{HASH}} + n(\lambda) \cdot q_{\mathsf{DE.Enc}}^{\text{HASH}} + q_A^{\text{HASH}}, \quad q_B^{\text{DEC}} = q_{S_1}^{\text{DEC}} + q_A^{\text{DEC}} ,$$

$$t_B \approx t_S + n(\lambda) \cdot t_{\mathsf{DE.Enc}} + t_A ,$$

$$q_D^{\text{HASH}} = n(\lambda) \cdot q_{\mathsf{DE.Enc}}^{\text{HASH}} + q_A^{\text{HASH}}, \quad q_D^{\text{DEC}} = q_A^{\text{DEC}} ,$$

$$t_D \approx n(\lambda) \cdot t_{\mathsf{DE.Enc}} + t_A .$$

$\underline{S_2.\mathsf{cx}(1^\lambda)}$
$cc \leftarrow\!\!\$\ S_1.\mathsf{cx}(1^\lambda)$ ; $j \leftarrow\!\!\$\ [n]$
Return $(cc, j)$
$\underline{S_2.\mathsf{msg}^{\text{Hash,Dec}}(1^\lambda, \overline{cc}, ek)}$
$(cc, j) \leftarrow \overline{cc}$
$\mathbf{m} \leftarrow S_1.\mathsf{msg}^{\text{Hash,Dec}}(1^\lambda, cc, ek)$
Return $\mathbf{m}[j]$

$\underline{B^{\text{Hash,Dec}}(1^\lambda, \overline{cc}, ek, c)}$
$(cc, j) \leftarrow \overline{cc}$
$\mathbf{m} \leftarrow\!\!\$\ S_1.\mathsf{msg}^{\text{Hash,Dec}}(1^\lambda, cc, ek)$
For $i \leftarrow 1, \ldots, n(\lambda)$ do
  $\mathbf{c}[i] \leftarrow\!\!\$\ \mathsf{DE.Enc}^{\text{Hash}}(1^\lambda, ek, \mathbf{m}[i])$
$\mathbf{c}[j] \leftarrow c$
$(m, i) \leftarrow A^{\text{Hash,DecSim}}(1^\lambda, cc, ek, \mathbf{c})$
If $(i = j)$ then Return $m$
Else Return $\perp$
$\underline{\text{Algorithm DecSim}(x)}$
If $(\exists i : x = \mathbf{c}[i])$ then return $\perp$
Return $\text{Dec}(x)$

$\underline{D^{\text{Hash,Dec}}(1^\lambda, cc, ek, \mathbf{m}, j)}$
For $i \leftarrow 1, \ldots, |\mathbf{m}|$ do $\mathbf{c}[i] \leftarrow \mathsf{DE.Enc}^{\text{Hash}}(1^\lambda, ek, \mathbf{m}[i])$
$(m, i) \leftarrow\!\!\$\ A^{\text{Hash,DecSim}}(1^\lambda, cc, ek, \mathbf{c})$
Return $\neg((\mathbf{m}[i] = m)$ and $(j = i))$
$\underline{\text{Algorithm DecSim}(x)}$
If $(\exists i : x = \mathbf{c}[i])$ then return $\perp$
Return $\text{Dec}(x)$

**Fig. 10.** Source $S_2$ (top left), adversary $B$ (top right), and distinguisher $D$ (bottom) used in Theorem 2.

---

*Furthermore, $S_2$ is unpredictable if $S_1$ is. Given any predictor $P_2$ for $S_2$, the proof gives predictor $P_1$ such that*

$$\mathsf{Adv}^{\mathsf{pred}}_{\mathsf{DE},S_2,P_2}(\lambda) \le \mathsf{Adv}^{\mathsf{pred}}_{\mathsf{DE},S_1,P_1}(\lambda)\ , \tag{16}$$

*and*

$$q^{\text{Hash}}_{S_2} = q^{\text{Hash}}_{S_1}, \quad q^{\text{Dec}}_{S_2} = q^{\text{Dec}}_{S_1}, \quad t_{S_2} \approx t_{S_1}\ ,$$
$$q^{\text{Hash}}_{P_2} = q^{\text{Hash}}_{P_1}, \quad q^{\text{Dec}}_{P_2} = q^{\text{Dec}}_{P_1}, \quad t_{P_2} \approx t_{P_1}\ .$$

The intuition behind the proof of Theorem 2 is straightforward—resampling-indistinguishability allows a PDMR adversary to *simulate* the ciphertext vector $\mathbf{c}$ in order to run *any* RI PDMR adversary. We give the details below.

*Proof (of Theorem 2).* Consider game $\text{G}_0$ and $\text{G}_1$ given in Fig. 11, where $\text{G}_1$ contains the boxed code, while $\text{G}_0$ does not. By construction,

$$\Pr[\text{G}_1] = \Pr[\mathbf{G}^{\text{pdmr}}_{\mathsf{DE},S_2,B}(\lambda)]\ . \tag{17}$$

Next, we claim that

$$\Pr[\text{G}_0] = \frac{1}{n(\lambda)} \cdot \Pr[\mathbf{G}^{\text{pdmr}}_{\mathsf{DE},S_1,A}(\lambda)]\ . \tag{18}$$

GAME $\mathrm{G}_0$ $\boxed{\mathrm{G}_1}$
$ek \leftarrow\!\!\$\ \mathsf{DE.Kg}(1^\lambda)$ ; $cc \leftarrow\!\!\$\ S_1.\mathsf{cx}(1^\lambda)$ ; $\mathbf{m} \leftarrow\!\!\$\ S_1.\mathsf{msg}^{\textsc{Hash},\textsc{Dec}}(1^\lambda, cc, ek)$
$j \leftarrow\!\!\$\ [n]$ ; $\boxed{\mathbf{m}[j] \leftarrow\!\!\$\ S_1.\mathsf{msg}^{\textsc{Hash}}(1^\lambda, cc, ek)[j]}$
For $i \leftarrow 1, \ldots, |\mathbf{m}|$ do $\mathbf{c}[i] \leftarrow \mathsf{DE.Enc}^{\textsc{Hash}}(1^\lambda, ek, \mathbf{m}[i])$
$(m, i) \leftarrow\!\!\$\ A^{\textsc{Hash},\textsc{Dec}}(1^\lambda, cc, ek, \mathbf{c})$ ; Return $((\mathbf{m}[i] = m)$ and $(j = i))$

$\textsc{Hash}(x, 1^\ell)$
If not $T[x, \ell]$ then $T[x, \ell] \leftarrow\!\!\$\ \{0,1\}^\ell$
Return $T[x, \ell]$

Algorithm $\textsc{Dec}(c)$
If $(\exists i : c = \mathbf{c}[i])$ then return $\perp$
Return $\mathsf{DE.Dec}(1^\lambda, dk, c)$

**Fig. 11.** Games $\mathrm{G}_0$ and $\mathrm{G}_1$ used in the proof of Theorem 2.

This is because $j$ is uniformly sampled and is not used any where in $\mathrm{G}_0$ besides computing the return value. Finally, let us consider $\mathbf{G}^{\mathrm{ri}}_{S_1,D}$. We note that by construction of $D$, it holds for $i \in \{0,1\}$ that

$$\Pr[\mathrm{G}_i] = \Pr[D \text{ outputs } 0 \mid b = i] \ , \tag{19}$$

where the second probability is taken over game $\mathbf{G}^{\mathrm{ri}}_{\mathsf{DE},S_1,D}$ and $b$ is as sampled in the game. Hence,

$$\Pr[\mathrm{G}_0] - \Pr[\mathrm{G}_1] = \mathsf{Adv}^{\mathsf{ri}}_{\mathsf{DE},S_1,D}(\lambda) \ . \tag{20}$$

Combining Equations (17), (18) and (20), we obtain Equation (15). Lastly, let $P_2$ be a predictor for $S_2$, consider the following predictor $P_1$ for $S_1$:

$\underline{P_1^{\textsc{Hash},\textsc{Dec}}(1^\lambda, \overline{cc}, ek, n, \boldsymbol{\ell})}$
$(cc, j) \leftarrow \overline{cc}$ ; $m \leftarrow P_2^{\textsc{Hash},\textsc{Dec}}(1^\lambda, cc, ek, 1, \boldsymbol{\ell}[j])$
Return $(m, j)$

It is easy to check that Equation (16) holds.

### 5.3 Security of EwH against $\mathcal{S}^{\mathsf{up}} \cap \mathcal{S}^{\mathsf{ri}}$

Combining Theorem 2 and Corollary 1, we obtain the following theorem, which says that if PKE is \$IND(-CPA) secure, then $\mathsf{DE} = \mathsf{EwH}[\mathsf{PKE}]$ is PDMR(-CPA)-secure against $\mathcal{S}^{\mathsf{ri}} \cap \mathcal{S}^{\mathsf{up}}$.

**Theorem 3.** *Let* PKE *be a public-key encryption scheme. Let* $\mathsf{DE} = \mathsf{EwH}[\mathsf{PKE}]$ *be the associated deterministic public-key encryption scheme. Let $S$ be a $n(\lambda)$-source for* DE*. Let $A$ be a PDMR adversary for $S$. \$IND adversary $B$ for* PKE*,*

*predictor $P$, and distinguisher $D$ can be constructed such that*
$\mathsf{Adv}^{\mathsf{pdmr}}_{\mathsf{DE},A}(\lambda) \leq n(\lambda) \cdot \mathsf{Adv}^{\mathsf{ri}}_{S,D}(\lambda)$

$$+ n(\lambda)\sqrt{\left(q_S^{\text{HASH}}+1\right)\left(q_S^{\text{HASH}}+q_A^{\text{HASH}}+n(\lambda)+1\right)\left(\mathsf{Adv}^{\$\mathsf{ind}}_{\mathsf{PKE},B}(\lambda)+\mathsf{Adv}^{\mathsf{pred}}_{S,P}(\lambda)\right)}\ .$$

*Furthermore, $D$, $B$ and $P$ are efficient as long as $S$ and $A$ are. In particular,*

$$q_B^{\text{HASH}} = 2\cdot q_S^{\text{HASH}} + q_A^{\text{HASH}} + n(\lambda) + 1, \quad q_B^{\text{DEC}} = 2\cdot q_S^{\text{DEC}} + q_A^{\text{DEC}}$$

$$t_B = 2\cdot t_S + n(\lambda)\cdot t_{\mathsf{PKE.Enc}} + t_A\ ,$$

$$q_D^{\text{HASH}} = n(\lambda) + q_A^{\text{HASH}},\ q_D^{\text{DEC}} = q_A^{\text{DEC}},\ t_D \approx n(\lambda)\cdot t_{\mathsf{DE.Enc}} + t_A\ ,$$

*and*

$$q_P^{\text{HASH}} = q_S^{\text{HASH}} + q_A^{\text{HASH}} + n(\lambda) + 1,\ q_P^{\text{DEC}} = q_S^{\text{DEC}} + q_A^{\text{DEC}},$$

$$t_P \approx t_S + n(\lambda)\cdot t_{\mathsf{PKE.Enc}} + t_A\ .$$

The proof of Theorem 3 is straight forward given Theorem 2 and Corollary 1 and we only sketch it here. We first apply Theorem 2 to source $S$ and adversary $A$ to obtain a 1-source $S'$, adversary $A'$ and distinguisher $D$. Then, we can apply Corollary 1 to $S'$ and $A'$ to obtain adversary $B$ and predictor $P$.

## 6 Impossibility Results

In this section, we explore what goes wrong when messages can have correlation. The known attacks showing IND-style security is unachievable [2, 5] only distinguish between encryptions of unpredictable messages. Here we give attacks showing that public-key-dependent messages can in fact be recovered in full by the adversary —that is, PDMR security is violated— as long as two or more *closely related* messages are encrypted. In particular, we show that no D-PKE scheme is secure against $\mathcal{S}^{\mathsf{up}}$ (in particular $\mathcal{S}^{\mathsf{up}} \cap \mathcal{S}^2$). We start with a basic attack on schemes that can encrypt messages of any length, and then extend this to schemes that can only encrypt messages of a fixed length.

Basic attack. The basic PDMR attack works when the D-PKE scheme allows the encryption of messages of arbitrary length, meaning $\mathsf{DE.IL}(\cdot) = \mathbb{N}$. The idea is simple. Since the message-choosing adversary $A_1$ has the public key, it can encrypt. It sets the second message to the encryption of a first, random message. The first challenge ciphertext is thus the second message. This requires that the scheme be able to encrypt messages of varying length because the ciphertext will not (usually) have the same length as the plaintext. For the attack to be valid, we must also show that the adversary is unpredictable. The following theorem formalizes this intuition. Here $\mu(\cdot)$ is a parameter representing the message length. The adversary is statistically unpredictable for $\mu(\cdot) = \omega(\log(\cdot))$, ruling out even weak PDMR security. The D-PKE scheme is *arbitrary* subject to being able to encrypt messages of arbitrary length.

| | |
|---|---|
| $\underline{S.\mathsf{cx}^{\text{HASH}}(1^\lambda)}$<br>Return $\epsilon$<br>$\underline{S.\mathsf{msg}^{\text{HASH}}(1^\lambda, \varepsilon, \mathsf{pp})}$<br>$\mathbf{m}[1] \leftarrow\!\!\$\ \{0,1\}^{\mu(\lambda)}$<br>$\mathbf{m}[2] \leftarrow \mathsf{DE.Enc}^{\text{HASH}}(1^\lambda, \mathsf{pp}, \mathbf{m}[1])$<br>Return $\mathbf{m}$<br><br>$\underline{A^{\text{HASH}}(1^\lambda, \varepsilon, \mathsf{pp}, \mathbf{c})}$<br>Return $(\mathbf{c}[1], 2)$ | $\underline{S.\mathsf{cx}^{\text{HASH}}(1^\lambda)}$<br>$hk \leftarrow\!\!\$\ \{0,1\}^{\mathsf{H.kl}}$<br>Return $hk$<br>$\underline{S.\mathsf{msg}^{\text{HASH}}(1^\lambda, hk, \mathsf{pp})}$<br>$\mathbf{m}[1] \leftarrow\!\!\$\ \{0,1\}^{\mu(\lambda)}$<br>$c \leftarrow \mathsf{DE.Enc}^{\text{HASH}}(1^\lambda, \mathsf{pp}, \mathbf{m}[1])$<br>$\mathbf{m}[2] \leftarrow \mathsf{H.Ev}(1^\lambda, hk, c)$<br>Return $\mathbf{m}$<br>$\underline{A^{\text{HASH}}(1^\lambda, hk, \mathsf{pp}, \mathbf{c})}$<br>Return $(\mathsf{H.Ev}(1^\lambda, hk, \mathbf{c}[1]), 2)$ |

**Fig. 12. Left:** Source $S$ and PDMR adversary $A$ used in Theorem 4. **Right:** Source $S$ and PDMR adversary $A$ used in Theorem 5.

---

**Theorem 4.** *Let* $\mathsf{DE}$ *be a D-PKE scheme with* $\mathsf{DE.IL}(\lambda) = \mathbb{N}$ *for all* $\lambda$*. Then,* $\mathsf{DE}$ *is not PDMR-secure against* $\mathcal{S}^{\mathsf{up}}$ *message samplers. In particular, let* $\mu\colon \mathbb{N} \to \mathbb{N}$ *be any function, and* $S, A$ *be the source and adversary given on the left in Fig. 12. Then, we claim that* $S \in \mathcal{S}^2 \cap \mathcal{S}^{\mathsf{up}}$*; in particular, for predictors* $P$ *and all* $\lambda$*,*

$$\mathsf{Adv}^{\mathsf{pred}}_{\mathsf{DE},S,P}(\lambda) \leq 2^{-\mu(\lambda)} \ . \tag{21}$$

*But for all* $\lambda$*,*

$$\mathsf{Adv}^{\mathsf{pdmr}}_{\mathsf{DE},S,A}(\lambda) = 1 \ . \tag{22}$$

*Proof (of Theorem 4).* We first prove Equation (22). Adversary $A$ wins game $\mathbf{G}^{\text{pdmr}}_{\mathsf{DE},A}(\lambda)$ as long as $\mathbf{m}[2]$, as computed by $A_1$, equals $\mathbf{c}[1]$, as computed by the game. Both are computed independently as $\mathsf{DE.Enc}^{\text{HASH}}(1^\lambda, \mathsf{pp}, \mathbf{m}[1])$, so they will always be equal, since $\mathsf{DE.Enc}$ is deterministic.

We move on to prove Equation (21). Let $P$ be any predictor, and consider game $\mathbf{G}^{\text{pred}}_{\mathsf{DE},S,P}(\lambda)$. For $i = 1, 2$ let $\mathrm{E}_i$ be the event that in game $\mathbf{G}^{\text{pred}}_{\mathsf{DE},S,P}(\lambda)$, predictor $P$ outputs a guess of the form $(m', i)$, for some string $m'$. The following inequalities, which complete the proof, are justified after they are stated:

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{pred}}_{\mathsf{DE},S,P}(\lambda) &= \sum_{i=1}^{2} \Pr[\, \mathbf{G}^{\text{pred}}_{\mathsf{DE},S,P}(\lambda) \,|\, \mathrm{E}_i \,] \cdot \Pr[\mathrm{E}_i] \\
&\leq 2^{-\mu(\lambda)} \cdot \Pr[\mathrm{E}_1] + 2^{-\mu(\lambda)} \cdot \Pr[\mathrm{E}_2] \qquad (23) \\
&\leq 2^{-\mu(\lambda)} \ . \qquad (24)
\end{aligned}$$

Since the first message $\mathbf{m}[1]$ is randomly chosen from $\{0,1\}^{\mu(\lambda)}$, the probability that $m' = \mathbf{m}[1]$ when $P$ returns $(m', 1)$ is at most $2^{-\mu(\lambda)}$. The second message

$\mathbf{m}[2]$ is the deterministic encryption of the first message, $\mathbf{m}[1]$. Since the function $\mathsf{DE.Enc}(1^\lambda, ek, \cdot)$ is injective, and there are $2^{\mu(\lambda)}$ possible values for $\mathbf{m}[1]$, there will also be $2^{\mu(\lambda)}$ possible values for $\mathbf{m}[2]$. So again, the probability that $m' = \mathbf{m}[2]$ when $P$ returns $(m', 2)$ is at most $2^{-\mu(\lambda)}$. This justifies Equation (23). Equation (24) holds simply because $\Pr[\mathrm{E}_1] + \Pr[\mathrm{E}_2] \leq 1$.

General attack. The basic attack assumed the D-PKE scheme could encrypt messages of varying length. Many D-PKE schemes —and even definitions— in the literature restrict the space of allowed messages to ones of a single length. We now extend the basic attack to one that works in this case, showing that no D-PKE scheme is (even weakly) PDMR-secure for the encryption of two or more messages, even if these are of the same length.

Function families. A family of functions (or function family) $\mathsf{F}$ specifies a deterministic PT evaluation algorithm $\mathsf{F.Ev}$ such that $\mathsf{F.Ev}(1^\lambda, \cdot, \cdot)$: $\{0,1\}^{\mathsf{F.kl}(\lambda)} \times \{0,1\}^{\mathsf{F.il}(\lambda)} \to \{0,1\}^{\mathsf{F.ol}(\lambda)}$ for all $\lambda \in \mathbb{N}$, where $\mathsf{F.kl}$, $\mathsf{F.il}$ and $\mathsf{F.ol}$ are the key, input and ouput length functions, respectively. Many security attributes may be defined and considered for such families.

Universal hash functions. As a tool we need a family of universal hash functions, so we start by recalling the definition. Let $\mathsf{H}$ be a family of functions. For $\lambda \in \mathbb{N}$, a key $hk \in \{0,1\}^{\mathsf{H.kl}(\lambda)}$ and inputs $x_1, x_2 \in \{0,1\}^{\mathsf{H.il}(\lambda)}$ we define the collision probabilities

$$\mathbf{cp}_{\mathsf{H}}(\lambda, x_1, x_2) = \Pr[\mathsf{H.Ev}(1^\lambda, hk, x_1) = \mathsf{H.Ev}(1^\lambda, hk, x_2)]$$
$$\mathbf{cp}_{\mathsf{H}}(\lambda) = \max\ \mathbf{cp}_{\mathsf{H}}(\lambda, x_1, x_2)\ ,$$

where the probability is over $hk \leftarrow\!\!\$\ \{0,1\}^{\mathsf{H.kl}(\lambda)}$ and the max is over all distinct $x_1, x_2 \in \{0,1\}^{\mathsf{H.il}(\lambda)}$. We say that $\mathsf{H}$ is *universal* if $\mathbf{cp}_{\mathsf{H}}(\lambda) = 2^{-\mathsf{H.ol}(\lambda)}$ for all $\lambda \in \mathbb{N}$.

**Theorem 5.** *Let* $\mathsf{DE}$ *be a D-PKE scheme. Let* $\mu\colon \mathbb{N} \to \mathbb{N}$ *be any function such that* $\mu(\lambda) \in \mathsf{DE.IL}(\lambda)$ *for all* $\lambda \in \mathbb{N}$. *We claim that* $\mathsf{DE}$ *is not PDMR-secure for* $\mathcal{S}^{\mathsf{up}}$ *message samplers. More precisely, let* $\mathsf{H}$ *be a universal family of functions with* $\mathsf{H.il}(\lambda) = \mathsf{DE.cl}(\lambda, \mu(\lambda))$ *and* $\mathsf{H.ol}(\lambda) = \mu(\lambda)$ *for all* $\lambda \in \mathbb{N}$. *Let* $S, A$ *be the source and PDMR adversary for* $\mathsf{DE}$ *shown on the right in Fig. 12. Then* $S \in \mathcal{S}^2 \cap \mathcal{S}^{\mathsf{up}}$*; in particular, for all predictors* $P$ *and all* $\lambda$*,*

$$\mathsf{Adv}^{\mathsf{pred}}_{\mathsf{DE},S,P}(\lambda) \leq \sqrt{2} \cdot 2^{-\mu(\lambda)/2}\ . \tag{25}$$

*But for all* $\lambda$*,*

$$\mathsf{Adv}^{\mathsf{pdmr}}_{\mathsf{DE},S,A}(\lambda) = 1\ . \tag{26}$$

The adversary picks $\mathbf{m}[1]$ as before and hashes its encryption down to get $\mathbf{m}[2]$. Note that both these strings have the same length $\mu(\lambda)$, so the attack works even if there is just one allowed message length. The key $hk$ for the hash function is shared using the common coins, so is available to both the message source and the adversary. The adversary continues to have PDMR advantage one. The more

difficult task is to establish its unpredictability. The theorem shows that the prediction advantage has degraded (increased) relative to Theorem 4, being about the square root of what it was before, but this is still exponentially vanishing with $\mu(\cdot)$. The proof of this bound uses techniques from the proof of the Leftover Hash Lemma [24].

*Proof (of Theorem 5).* We first prove Equation (26). Adversary $A$ wins game $\mathbf{G}^{\mathrm{pdmr}}_{\mathsf{DE},S,A}(\lambda)$ as long as $\mathbf{m}[2]$ from $S$ equals $\mathsf{H.Ev}(1^\lambda, hk, \mathbf{c}[1])$, as computed by the game. Both are calculated as $\mathsf{H.Ev}(1^\lambda, hk, \mathsf{DE.Enc}^{\mathrm{Hash}}(1^\lambda, \mathsf{pp}, \mathbf{m}[1]))$, so they will always be equal, since $\mathsf{DE.Enc}$ is deterministic.

Now we prove Equation (25). Let $P$ be any predictor, and consider game $\mathbf{G}^{\mathrm{pred}}_{\mathsf{DE},S,P}(\lambda)$. For $i = 1, 2$ let $\mathrm{E}_i$ be the event that in game $\mathbf{G}^{\mathrm{pred}}_{\mathsf{DE},S,P}(\lambda)$, predictor $P$ outputs a guess of the form $(m', i)$, for some string $m'$. We claim that

$$\Pr[\,\mathbf{G}^{\mathrm{pred}}_{A,P}(\lambda) \,|\, \mathrm{E}_1\,] \le 2^{-\mu(\lambda)} \tag{27}$$

$$\Pr[\,\mathbf{G}^{\mathrm{pred}}_{A,P}(\lambda) \,|\, \mathrm{E}_2\,] \le \sqrt{2} \cdot 2^{-\mu(\lambda)/2}\ . \tag{28}$$

Given the above, we can complete the proof via

$$\begin{aligned}\mathsf{Adv}^{\mathsf{pred}}_{\mathsf{DE},S,P}(\lambda) &= \sum_{i=1}^{2} \Pr[\,\mathbf{G}^{\mathrm{pred}}_{\mathsf{DE},S,P}(\lambda) \,|\, \mathrm{E}_i\,] \cdot \Pr[\mathrm{E}_i] \\ &\le 2^{-\mu(\lambda)} \cdot \Pr[\mathrm{E}_1] + \sqrt{2} \cdot 2^{-\mu(\lambda)/2} \cdot \Pr[\mathrm{E}_2] \\ &\le \sqrt{2} \cdot 2^{-\mu(\lambda)/2}\end{aligned}$$

Equation (27) is true for the same reason as in Theorem 4, namely that, since the first message $\mathbf{m}[1]$ is randomly chosen from $\{0,1\}^{\mu(\lambda)}$, the probability that $m' = \mathbf{m}[1]$ when $P$ returns $(m', 1)$ is at most $2^{-\mu(\lambda)}$. The main issue is Equation (28), which we now prove.

Let $(ek, dk) \in [\mathsf{DE.Kg}(1^\lambda)]$ and $hk \in \{0,1\}^{\mathsf{H.kl}(\lambda)}$. Define $\mathsf{X}_{ek,hk}\colon\ \{0,1\}^{\mu(\lambda)} \to \{0,1\}^{\mu(\lambda)}$ by

$$\mathsf{X}_{ek,hk}(m) = \mathsf{H.Ev}(1^\lambda, hk, \mathsf{DE.Enc}(1^\lambda, ek, m))\ .$$

Regard this as a random variable over the random choice of $m \leftarrow\!\!\$\ \{0,1\}^{\mu(\lambda)}$. Now consider the guessing and collision probabilities of this random variable,

$$\mathbf{gp}(\mathsf{X}_{ek,hk}) = \max_{h \in \{0,1\}^{\mathsf{H.ol}(\lambda)}} \Pr[\mathsf{X}_{ek,hk} = h]$$

$$\mathbf{cp}(\mathsf{X}_{ek,hk}) = \sum_{h \in \{0,1\}^{\mathsf{H.ol}(\lambda)}} \Pr[\mathsf{X}_{ek,hk} = h]^2\ .$$

Further define $\mathsf{GP}_{ek}, \mathsf{CP}_{ek}\colon\ \{0,1\}^{\mathsf{H.kl}(\lambda)} \to [0,1]$ by

$$\mathsf{GP}_{ek}(hk) = \mathbf{gp}(\mathsf{X}_{ek,hk}) \quad \text{and} \quad \mathsf{CP}_{ek}(hk) = \mathbf{cp}(\mathsf{X}_{ek,hk})\ ,$$

and regard them as random variables over the random choice of $hk \leftarrow\!\!\$\ \{0,1\}^{\mathsf{H.kl}(\lambda)}$. Below we will show that

$$\mathbf{E}\,[\,\mathsf{GP}_{ek}\,] \le \sqrt{2} \cdot 2^{-\mu(\lambda)/2} \tag{29}$$

for every $(ek, dk) \in [\mathsf{DE.Kg}(1^\lambda)]$. Now, $hk$ is an input to $P$, so

$$\begin{aligned}\Pr[\,\mathbf{G}^{\mathrm{pred}}_{A,P}(\lambda) \mid \mathrm{E}_2\,] &\leq \max_{(ek,dk)\in[\mathsf{DE.Kg}(1^\lambda)]} \mathbf{E}\,[\,\mathsf{GP}_{ek}\,] \\ &\leq \sqrt{2}\cdot 2^{-\mu(\lambda)/2}\end{aligned}$$

where the second equation is by Equation (29). This proves Equation (28). Fixing $(ek, dk) \in [\mathsf{DE.Kg}(1^\lambda)]$, we now prove Equation (29). It is clear (and a standard relation between guessing and collision probabilities of a random variable) that for all $hk$ we have

$$\mathbf{gp}(\mathsf{X}_{ek,hk})^2 \leq \mathbf{cp}(\mathsf{X}_{ek,hk})\ .$$

Thus

$$\mathsf{GP}_{ek} \leq \sqrt{\mathsf{CP}_{ek}}\ .$$

By Jensen's inequality and concavity of the square-root function,

$$\mathbf{E}\,[\,\mathsf{GP}_{ek}\,] \leq \mathbf{E}\left[\,\sqrt{\mathsf{CP}_{ek}}\,\right] \leq \sqrt{\mathbf{E}\,[\,\mathsf{CP}_{ek}\,]}\ .$$

Now with the expectation over $hk \leftarrow\!\!\$\ \{0,1\}^{\mathsf{H.kl}(\lambda)}$ and the probability over $m_1$, $m_2 \leftarrow\!\!\$\ \{0,1\}^{\mu(\lambda)}$, we have

$$\mathbf{E}\,[\,\mathsf{CP}_{ek}\,] = \mathbf{E}\,[\,\Pr[\mathsf{X}_{ek,hk}(m_1) = \mathsf{X}_{ek,hk}(m_2)]\,] \leq 2^{-\mu(\lambda)} + \mathbf{cp}_{\mathsf{H}}(\lambda)\ .$$

This is by considering two cases. The first is that $m_1 = m_2$, which happens with probability $2^{-\mu(\lambda)}$. The second is that $m_1 \neq m_2$, in which case the inputs to $\mathsf{H.Ev}(1^\lambda, hk, \cdot)$ are different due to the injectivity of $\mathsf{DE.Enc}(1^\lambda, ek, \cdot)$, and we can exploit the universality of $\mathsf{H}$. Now by assumption of universality of $\mathsf{H}$, $\mathbf{cp}_{\mathsf{H}}(\lambda) = 2^{-\mu(\lambda)}$, so putting everything together we have Equation (29).

## Acknowledgments

The first and second authors are supported in part by NSF grants CNS-1526801 and CNS-1717640, ERC Project ERCC FP7/615074 and a gift from Microsoft. The second author is supported in part by a Powell fellowship. The third author was supported in part by NSF grant CNS-1564102.

We thank reviewers from Asiacrypt 2019 and Crypto 2019 for their detailed and extensive comments.

## References

1. A. Bagherzandi, J. H. Cheon, and S. Jarecki. Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma. In P. Ning, P. F. Syverson, and S. Jha, editors, *ACM CCS 2008*, pages 449–458. ACM Press, Oct. 2008. 2, 4
2. M. Bellare, A. Boldyreva, and A. O'Neill. Deterministic and efficiently searchable encryption. In A. Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 535–552. Springer, Heidelberg, Aug. 2007. 1, 2, 3, 4, 11, 12, 15, 24

3. M. Bellare, W. Dai, and L. Li. The local forking lemma and its application to deterministic encryption. Cryptology ePrint Archive, Report 2019/1017, 2019. https://eprint.iacr.org/2019/1017. 5
4. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In H. Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 26–45. Springer, Heidelberg, Aug. 1998. 5
5. M. Bellare, M. Fischlin, A. O'Neill, and T. Ristenpart. Deterministic encryption: Definitional equivalences and constructions without random oracles. In D. Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 360–378. Springer, Heidelberg, Aug. 2008. 1, 2, 11, 24
6. M. Bellare and V. T. Hoang. Resisting randomness subversion: Fast deterministic and hedged public-key encryption in the standard model. In E. Oswald and M. Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 627–656. Springer, Heidelberg, Apr. 2015. 1, 5
7. M. Bellare and G. Neven. Multi-signatures in the plain public-key model and a general forking lemma. In A. Juels, R. N. Wright, and S. De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 390–399. ACM Press, Oct. / Nov. 2006. 2, 3, 4, 7, 8, 10
8. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In D. E. Denning, R. Pyle, R. Ganesan, R. S. Sandhu, and V. Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, Nov. 1993. 6
9. M. Bellare and P. Rogaway. Optimal asymmetric encryption. In A. D. Santis, editor, *EUROCRYPT'94*, volume 950 of *LNCS*, pages 92–111. Springer, Heidelberg, May 1995. 4, 15
10. M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Heidelberg, May / June 2006. 6
11. D. J. Bernstein, T. Lange, and R. Niederhagen. Dual EC: A standardized back door. Cryptology ePrint Archive, Report 2015/767, 2015. http://eprint.iacr.org/2015/767. 1
12. D. Bleichenbacher. On the security of the KMOV public key cryptosystem. In B. S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 235–248. Springer, Heidelberg, Aug. 1997. 4
13. A. Boldyreva, S. Fehr, and A. O'Neill. On notions of security for deterministic encryption, and efficient constructions without random oracles. In D. Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 335–359. Springer, Heidelberg, Aug. 2008. 1, 4, 5
14. Z. Brakerski and G. Segev. Better security for deterministic public-key encryption: The auxiliary-input setting. In P. Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 543–560. Springer, Heidelberg, Aug. 2011. 1, 5
15. D. R. L. Brown. A weak-randomizer attack on RSA-OAEP with e = 3. Cryptology ePrint Archive, Report 2005/189, 2005. http://eprint.iacr.org/2005/189. 1
16. R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In H. Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 13–25. Springer, Heidelberg, Aug. 1998. 5
17. R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003. 4
18. E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In M. J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 537–554. Springer, Heidelberg, Aug. 1999. 4, 15

19. B. Fuller, A. O'Neill, and L. Reyzin. A unified approach to deterministic encryption: New constructions and a connection to computational entropy. *Journal of Cryptology*, 28(3):671–717, July 2015. 1, 5
20. S. Garg, R. Gay, and M. Hajiabadi. New techniques for efficient trapdoor functions and applications. In Y. Ishai and V. Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 33–63. Springer, Heidelberg, May 2019. 1, 5
21. R. Gay, D. Hofheinz, E. Kiltz, and H. Wee. Tightly CCA-secure encryption without pairings. In M. Fischlin and J.-S. Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 1–27. Springer, Heidelberg, May 2016. 4
22. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984. 5
23. D. Hofheinz and E. Kiltz. Practical chosen ciphertext secure encryption from factoring. In A. Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 313–332. Springer, Heidelberg, Apr. 2009. 4
24. R. Impagliazzo and D. Zuckerman. How to recycle random bits. In *30th FOCS*, pages 248–253. IEEE Computer Society Press, Oct. / Nov. 1989. 5, 27
25. K. Kurosawa and Y. Desmedt. A new paradigm of hybrid encryption scheme. In M. Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 426–442. Springer, Heidelberg, Aug. 2004. 4
26. S. Micali, C. Rackoff, and B. Sloan. The notion of security for probabilistic cryptosystems. *SIAM Journal on Computing*, 17(2):412–426, Apr. 1988. Special issue on cryptography. 5
27. I. Mironov, O. Pandey, O. Reingold, and G. Segev. Incremental deterministic public-key encryption. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 628–644. Springer, Heidelberg, Apr. 2012. 1, 5
28. K. Ouafi and S. Vaudenay. Smashing SQUASH-0. In A. Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 300–312. Springer, Heidelberg, Apr. 2009. 1
29. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, June 2000. 2, 3, 4
30. A. Raghunathan, G. Segev, and S. P. Vadhan. Deterministic public-key encryption for adaptively chosen plaintext distributions. In T. Johansson and P. Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 93–110. Springer, Heidelberg, May 2013. 1, 2, 3, 5, 21
31. S. Yilek, E. Rescorla, H. Shacham, B. Enright, and S. Savage. When private keys are public: results from the 2008 debian openssl vulnerability. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement*, pages 15–27. ACM, 2009. 1