Enhanced Chosen-Ciphertext Security and Applications

Dana Dachman-Soled[1] Georg Fuchsbauer[2] Payman Mohassel[3]
Adam O'Neill[4]

Abstract

We introduce and study a new notion of *enhanced chosen-ciphertext security* (ECCA) for public-key encryption. Loosely speaking, in the ECCA security experiment, the decryption oracle provided to the adversary is augmented to return not only the output of the decryption algorithm on a queried ciphertext but also of a *randomness recovery* algorithm associated to the scheme. Our results mainly concern the case where the randomness recovery algorithm is efficient.

We provide constructions of ECCA-secure encryption from adaptive trapdoor functions as defined by Kiltz *et al.* (EUROCRYPT 2010), resulting in ECCA encryption from standard number-theoretic assumptions. We then give two applications of ECCA-secure encryption: (1) We use it as a unifying concept in showing equivalence of adaptive trapdoor functions and tag-based adaptive trapdoor functions, resolving an open question of Kiltz *et al.* (2) We show that ECCA-secure encryption can be used to securely realize an approach to public-key encryption with non-interactive opening (PKENO) originally suggested by Damgård and Thorbek (EUROCRYPT 2007), resulting in new and practical PKENO schemes quite different from those in prior work.

Our results demonstrate that ECCA security is of both practical and theoretical interest.

[1] Department of Electrical and Computer Engineering, University of Maryland, 3407 A.V. Williams Building, College Park, MD 20742, USA. Email: danadach@ece.umd.edu. URL: http://ece.umd.edu/danadach.

[2] Institute of Science and Technology Austria, Am Campus 1, 3400 Klosterneuburg, Austria. Email: georg.fuchsbauer@ist.ac.at. URL: http://www.di.ens.fr/~fuchsbau. Work started when at Bristol University, supported by EPSRC via grant EP/H043454/1.

[3] Department of Computer Science, University of Calgary, 2500 University Dr. NW, Calgary, AB. Email: pmohasse@cpsc.ucalgary.ca. URL: http://pages.cpsc.ucalgary.ca/~pmohasse.

[4] Department of Computer Science, Georgetown University, 3700 Reservoir Road NW Washington, DC 20057. Email: adam@cs.georgetown.edu. URL: http://cs.georgetown.edu/adam. Supported in part by NSF grants 0546614, 0831281, 1012910, and 1012798.

Contents

1 Introduction

This paper introduces and studies a new notion of security for public-key encryption (PKE) we call *enhanced* chosen-ciphertext security (ECCA). Besides being interesting in its own right, we find that ECCA security plays a fundamental role in contexts where randomness-recovering encryption (as discussed informally in e.g. [36]) is important, such as adaptive trapdoor functions [29] and PKE with non-interactive opening [15] (which is useful in secure multiparty computation). We also believe ECCA will find further applications in the future. Below we describe our results concerning ECCA in more detail; for a pictorial summary, see Figure 1.

1.1 ECCA Security Definition and Variants

Recall that in the standard formulation of CCA security [37], the adversary, given a public key pk, must guess which of the two possible messages its challenge ciphertext c encrypts, while being allowed to query a decryption oracle on any ciphertext c' different from c. Very informally, our "enhancement" is that the decryption oracle, when queried on a ciphertext c', returns not only the output of the decryption algorithm of the scheme run on c', but also of an associated *randomness-recovery* algorithm. This randomness-recovery algorithm, given sk and an honestly generated encryption c of m with coins r, is guaranteed to output some coins r' such that the encryption of m with coins r' is also c. (However, like the decryption algorithm — which is only guaranteed to output the right message on honestly generated ciphertexts — its behavior on other, maliciously generated ciphertexts depends on its specification.)

Note that in general we do not require that $r = r'$ above, but in the special case that this holds we say that the scheme is *uniquely* randomness-recovering. Looking ahead, our constructions of ECCA-secure PKE will achieve unique randomness-recoverability, but for some applications this is not strictly necessary as long as the scheme has perfect correctness (i.e., zero decryption error).

Our study of ECCA security is largely motivated by the related concept of randomness-recovering encryption, in which case the randomness-recovery algorithm is efficient. Indeed, we show that not every CCA-secure randomness-recovering encryption scheme is ECCA-secure (cf. Proposition 3.2). This means that in applications of randomness-recovering encryption that require ECCA security, it may not be sufficient to use a scheme proven CCA-secure.

1.2 Constructions of ECCA-Secure PKE

ECCA-Secure PKE from "adaptive" TDFs. The first standard-model construction of CCA-secure randomness-recovering PKE was achieved by Peikert and Waters [36], based on their new concept of "lossy" trapdoor functions (TDFs). A line of subsequent work [38, 29] focused on achieving CCA-secure PKE from progressively weaker assumptions on TDFs.[1] This leads one to wonder whether these assumptions suffice for ECCA-secure randomness-recovering PKE as well. Ideally, one would achieve ECCA-secure, uniquely randomness-recovering PKE — the strongest form of randomness-recovery — based on adaptive TDFs, the weakest of these assumptions. (Intuitively, adaptivity is a form of CCA security for TDFs, asking that the TDF remain one-way even when the adversary may query an inversion oracle on points other than its challenge.) This is exactly what our results obtain.

Challenges and techniques. Our construction is technically novel, as the construction of CCA-secure encryption from adaptive TDFs in the earlier work of [29] seems to be neither randomness-recovering nor ECCA-secure (we achieve both, and moreover *unique* randomness recovery). Indeed, in the construction of [29] a general transform of [33] is used to convert a one-bit CCA-secure PKE from ATDFs to a multi-bit CCA-secure one. However, this transform does not seem to preserve either randomness-recovery nor

[1] Wee [40] showed that a weaker notion of adaptivity for *trapdoor relations* suffices; however, as this is not an assumption on trapdoor *functions* it does seem to yield randomness-recovering encryption and won't be useful for our results.

ECCA-security of the one-bit scheme. Furthermore, the one-bit scheme of [29] — which works by re-sampling a domain point x until the hardcore bit of x equals the message— is not uniquely randomness-recovering, since decryption does not recover the "thrown away" x's. (Note that the "naïve" one-bit scheme from ATDFs that simply XOR's the message bit with a hardcore bit of the ATDF is trivially malleable by flipping the last bit of a ciphertext and thus is *not* CCA-secure.)

We solve these problems via a novel application of *detectable CCA (DCCA) security*, introduced recently by Hohenberger *et al.* [27]. Informally, DCCA is defined relative to a "detecting" function $\mathcal{F}$ that determines whether two ciphertexts are related; in the DCCA experiment, the adversary is not allowed to ask for decryptions of ciphertexts related to the challenge ciphertext according to $\mathcal{F}$. The work of [27] gives a transform from any DCCA-secure PKE to a CCA-secure encryption one. (For the transform to work, $\mathcal{F}$ must satisfy some conditions discussed in Section 4.2.1.) In particular, bit-by-bit encryption using a 1-bit CCA-secure encryption scheme is DCCA-secure, thus encompassing the earlier work of [33]. Our novelty is that we construct a DCCA-secure scheme from ATDFs by using the "naïve" one-bit scheme described above. We show this one-bit scheme is uniquely randomness-recovering and moreover satisfies a notion of DCCA with analogous "enhanced" security (where the decryption oracle also returns coins). We observe that (enhanced) DCCA (unlike CCA) is preserved under parallel composition (with a suitable change to the detecting function), so we can easily get a multi-bit scheme as well. Finally, to go all the way to ECCA-security, we show that the transform of [27] (in contrast to that of [33]) preserves both this enhanced security as well as (unique) randomness recovery for the resulting scheme. We thus get uniquely randomness-recovering ECCA-secure PKE from ATDFs as desired.

More efficient schemes. We note that the above is a feasibility result in terms of minimal assumptions. We also show more efficient constructions of ECCA-secure encryption from *tag-bsaed* ATDFs as defined in [29] and from ATDFs having a large number of simultaneous hardcore bits (using the KEM/DEM paradigm). See Section 4.3 and Appendix B for details.

1.3 Applications to Adaptive Trapdoor Functions

Going the other direction, we next give applications of ECCA-security to the theory of adaptive ATDFs. Namely, we show (1) adaptive TDFs are in fact *equivalent* to *uniquely* randomness-recovering ECCA-secure PKE. This helps us better understand the power and complexity of adaptive TDFs. We furthermore show (2) "tag-based" ATDFs as defined in [29] are likewise equivalent to uniquely randomness-recovering ECCA-secure PKE. A corollary of (1) and (2) is that tag-based and non-tag-based ATDFs are themselves equivalent, which resolves a foundational question left open by [29]. We note that it is in fact much easier to construct uniquely randomness-recovering ECCA-secure PKE from tag-based ATDFs than from non-tag-based ATDFs. (The rough intuition is that in the tag-based case, a signature scheme can be used to "glue together" many one-bit encryptions via a common tag, namely a single verification key.) Indeed, the apparent extra power of tag-based ATDFs makes it surprising that they turn out to be equivalent to (non-tag-based) ATDFs. We note that unlike the TDF case, the equivalence of tag-based and standard PKE is much easier to prove [30].

1.4 Applications to Public-Key Encryption with Non-Interactive Opening

Background. Public-key encryption with non-interactive opening (PKENO), introduced by Damgård an Thorbeck [16] and studied in detail by [15, 19, 20], allows a receiver to non-interactively prove to anyone that a ciphertext c decrypts to a message m. As discussed in the above-mentioned work, PKENO has applications to multiparty computation (*e.g.*, auctions and elections), secure message transmission, group signatures, and more. But despite numerous applications, such schemes have been difficult to realize. Secure constructions of PKENO currently exist from identity-based encryption [15] and robust non-interactive threshold encryption [20], which are somewhat heavy-weight primitives.

Resurrecting a simple approach. We show that ECCA-secure encryption can be used to securely realize (for the first time) a simple approach to PKENO originally suggested by [16]. The basic idea is to use a randomness-recovering PKE and have the receiver provide the recovered coins as the proof. However, several issues need to be addressed for this approach to work. One problem already discussed in [20, Section 4.1] is that there must also be a way for the receiver to prove the claimed behavior of the decryption algorithm on ciphertexts that might not even be an output of the encryption algorithm, and for which no underlying coins necessarily exist. (Note that such ciphertexts may or may not decrypt to $\perp$ in general.) More fundamentally, we observe that the encryption scheme *must be ECCA secure* (which was not even defined in prior work); standard chosen-ciphertext security is not enough here, because here the adversary in the corresponding PKENO security game has the ability to see random coins underlying ciphertexts of its choosing. We now describe our results in more detail.

PKENO-compatible ECCA encryption. First, we formalize a notion of *PKENO-compatible* ECCA-secure encryption, for which we can overcome the above problems and safely use the underlying message and randomness as the non-interactive opening of a ciphertext. There are two requirements for such a scheme: (1) It has a "partial-randomness" recovery algorithm that, informally, recovers enough coins to uniquely identify the underlying message. (Here "full" randomness-recovery is not needed, and would not permit constructions where the "randomized" part of the ciphertext is still verifiable, like a one-time signature or zero-knowledge proof.) This should also be true for ciphertexts *outside* the range of the encryption algorithm but which do not decrypt to $\perp$.[2] (2) The scheme has *ciphertext verifiability*, meaning one can check without the secret key (but possibly with the help of the recovered partial coins) whether the decryption of a ciphertext is $\perp$. Note that ECCA security of such schemes is defined with respect to the partial-randomness recovery algorithm.

We also define an analogous notion of PKENO-compatible ECCA-secure *tag-based* PKE. We show that one can efficiently transform such a scheme into a (non-tag-based) PKENO-compatible ECCA-secure PKE scheme using either of the two "BCHK transforms" [8]. (Recall that [8] give a "basic" transform using one-time signatures and a "more efficient" transform based on symmetric-key primitives.)

PKENO from ECCA+NIZK. We next show a generic way to achieve PKENO-compatibility from any ECCA-secure randomness-recovering PKE by adding a non-interactive zero-knowledge (NIZK) proof of "well-formedness" to a ciphertext, namely that exist some underlying message and random coins. (Indeed, the idea of adding such a proof to achieve PKENO comes from [16, 20], although not in connection with ECCA.) For the security proof to go through, the PKE scheme does not need to have unique randomness recovery, but it needs perfect correctness. Moreover, we show the NIZK needs to be simulation-sound, for reasons analogous to the proof of full anonymity of the group signature construction in [4].

Efficient PKENO-compatible tag-based PKE. The above construction is generic but inefficient. Towards more efficient schemes, we show our construction of ECCA-secure *tag-based* PKE from tag-based ATDFs can be made PKENO-compatible if its starting tag-based ATDF has "range verifiability" (i.e., anyone can verify preimage existence of a range point). We two efficient such tag-based ATDFs. The first instantiates a general tag-based ATDF construction from [29] that combines a lossy and all-but-one TDF as defined in [36]. Specifically, we use the lossy and all-but-one TDFs of Freeman *et al.* [18] based on the decision-linear (DLIN) assumption. We show that in this case preimage existence is a "Groth-Sahai" statement [25], for which we know efficient NIZK constructions in bilinear groups.[3] Interestingly, we show simulation-soundness is not needed in this case, illustrating another efficiency benefit over the generic approach. The second is a tag-based ATDF from [29] based on the "instance-independent" RSA assumption (II-RSA), which we observe intrinstically has range verifiability because it is a *permutation*.

[2] For example, consider a randomness-recovering scheme which always outputs ciphertexts whose last bit is "0," but whose decryption algorithm ignores this last bit. Then clearly we can still recover the randomness underlying ciphertexts whose last bit is "1" despite the fact that such ciphertexts are outside the range of the encryption algorithm.

[3] Technically, when the NIZK is added, the tag-based ATDF is not a trapdoor function anymore is already a tag-based PKE scheme (because the NIZK part is randomized), but we gloss over this technicality in our informal exposition.

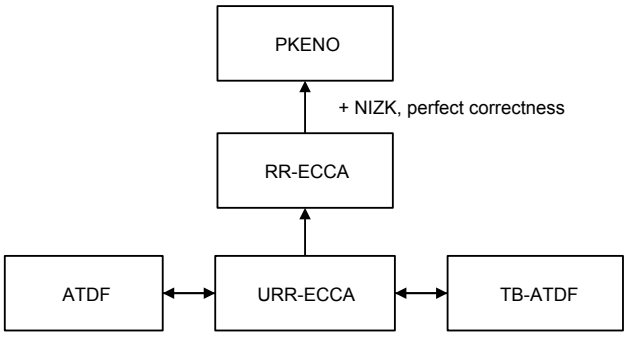

Figure 1: Relations between various primitives studied in this paper. "(U)RR-ECCA" is (uniquely) randomness-recovering enhanced-chosen-ciphertext secure PKE. "ATDF" is adaptive trapdoor function. "TB-ATDF" is tag-based adaptive trapdoor function. "PKENO" is public-key encryption with non-interactive opening.

The resulting PKENO scheme based on II-RSA is quite practical (see Section 7.2 for details).

1.5 Related Work

ECCA is similar in spirit to *coin-revealing selective opening attack* (SOA-C) [9, 17, 3, 7]. In the latter setting, there are say n ciphertexts encrypting related (but unknown) messages under independent random coins, and the adversary requests the plaintexts and random coins corresponding to some subset of them; the question is whether the "unopened" ciphertexts remain secure. However, it seems to us that SOA-C is neither implied by, nor implies, ECCA. It is an interesting question whether ECCA has any applications in the domain of SOA-C.

An analogue of ECCA (in the case of inefficient randomness-recovery) has been previously defined for *commitment schemes* by Canetti *et al.* [11], which they call CCA-secure commitments. These are commitment schemes that remain secure when the adversary has access to an *unbounded decommitment oracle* that it can call on commitments other than the challenge. They are interested in such schemes that are interactive but in the plain model, meaning there are no public keys. Thus, our setting seems incomparable (as we disallow interaction but allow public keys). However, we view their work as supporting the claim that ECCA is a natural notion of security to consider for encryption.

Other variants of CCA-security for encryption considered before include *replayable* CCA security [10], *constrained* CCA security [26], and *detectable* CCA security [27]. Notably, these are all *relaxations* of CCA security, whereas we consider a strengthening. Another strengthening of CCA security previously considered is *plaintext awareness* [6, 2, 5].

2 Preliminaries

2.1 Notation and Conventions

If A is an algorithm then $y \leftarrow A(x_1, \ldots, x_n; r)$ means we run A on inputs $x_1, \ldots, x_n$ and coins r and denote the output by y. By $y \leftarrow\!\!\$\, A(x_1, \ldots, x_n)$ we denote the operation of picking r at random and letting $y \leftarrow A(x_1, \ldots, x_n; r)$. Unless otherwise indicated, an algorithm may be randomized. "PPT" stands for "probabilistic polynomial time" and "PT" stands for "polynomial time." The security parameter is denoted $k \in \mathbb{N}$. If we say that an algorithm is efficient we mean that it is PPT (in the security parameter). All algorithms we consider are efficient unless indicated otherwise.

2.2 Public-key Encryption

A *public-key encryption scheme* [24] with message-space $MsgSp$ is a triple of algorithms $\mathsf{PKE} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$. The key-generation algorithm Kg returns a public key pk and matching secret key sk. The encryption algorithm Enc takes pk and a plaintext m to return a ciphertext. The deterministic decryption algorithm Dec takes sk and a ciphertext c to return a plaintext.

Correctness. An issue that will be more important than usual in our context is *correctness*, which refers to how likely it is that an encrypted message decrypts to some other message. By default we require *perfect correctness*: for all $k \in \mathbb{N}$ and $m \in MsgSp(1^k)$,

$$\Pr[\mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) = m \ : \ (pk, sk) \leftarrow\!\!\$\ \mathsf{Kg}(1^k)]$$

is 1. If instead we allow this probability to be $1 - \nu(k)$ we say that that PKE has *decryption error* $\nu(\cdot)$.

Tag-based. We say that PKE is *tag-based* [31] with tag-space $TagSp$ if $\mathsf{Enc}, \mathsf{Dec}$ take an additional input $t \in TagSp(1^k)$ called the *tag*. Again, by default we require *perfect correctness*: for all $k \in \mathbb{N}$, $m \in MsgSp(1^k)$, and $t \in TagSp(1^k)$,

$$\Pr[\mathsf{Dec}(sk, t, \mathsf{Enc}(pk, t, m)) - m \ : \ (pk, sk) \leftarrow\!\!\$\ \mathsf{Kg}(1^k)]$$

is 1. Decryption error is defined analogously.

Other standard primitives. We recall the definitions of other standard primitives such as (injective) trapdoor functions in Appendix A.

3 Enhanced Chosen-Ciphertext Security

Randomness recovery. We start with a definition of *randomness recovery* for public-key encryption. For any public-key encryption scheme $\mathsf{PKE} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ we specify an additional *randomness recovery* algorithm that takes a secret key sk and ciphertext c to return coins r; that is, we write $\mathsf{PKE} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Rec})$. To our knowledge, this notion has been discussed informally in the literature (*e.g.* in [36]) but our formalization is novel. Suppose Enc draws its coins from *Coins*. We require that for all messages $m \in MsgSp(1^k)$

$$\Pr[\mathsf{Enc}(pk, m; r') \neq c \ : \ (pk, sk) \leftarrow\!\!\$\ \mathsf{Kg}\ ;\ r \leftarrow\!\!\$\ Coins(1^k)\ ;\ c \leftarrow \mathsf{Enc}(pk, m; r)\ ;\ r' \leftarrow \mathsf{Rec}(sk, c)]$$

is negligible. Note that we do *not* necessarily require $r = r'$; that is, the randomness recovery algorithm need not return the *same* coins used for encryption; indeed, it may not be possible, information theoretically, to determine r from sk and c. We also do not require Rec to be efficient in general. But in the special case that Rec is PT we say that PKE is *randomness recovering*. Moreover, if the forgoing condition on Rec holds for $r = r'$ we say that PKE is *uniquely* randomness recovering.[4] In the definition that follows these are important special cases, but they are not assumed by the definition.

In the case of tag-based public-key encryption, Rec also takes a *tag* as input. In this case we require that for all $m \in MsgSp(1^k)$ and $t \in TagSp(1^k)$

$$\Pr[\mathsf{Enc}(pk, t, m; r') \neq c \ : \ (pk, sk) \leftarrow\!\!\$\ \mathsf{Kg}\ ;\ r \leftarrow\!\!\$\ Coins(1^k)\ ;\ c \leftarrow \mathsf{Enc}(pk, t, m; r)\ ;\ r' \leftarrow \mathsf{Rec}(sk, t, c)]$$

is negligible. Randomness-recovery and unique randomness-recovery are defined analogously.

ECCA definition. We are now ready to state our new definition. Let $\mathsf{PKE} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme. We associate to PKE and an adversary $A = (A_1, A_2)$ an *enhanced chosen-ciphertext attack* experiment,

[4] Looking ahead, it turns out that in some applications of ECCA, non-unique randomness recovery is OK as long as the scheme has perfect correctness.

Experiment $\mathbf{Exp}^{\text{ind-ecca}}_{\mathsf{PKE},A}(k)$
$b \leftarrow\!\!\$\ \{0,1\}$; $(pk, sk) \leftarrow\!\!\$\ \mathsf{Kg}(1^k)$
$(m_0, m_1, St) \leftarrow\!\!\$\ A_1^{\mathsf{Dec}^*(sk,\cdot)}(pk)$
$c \leftarrow\!\!\$\ \mathsf{Enc}(pk, m_b)$
$d \leftarrow\!\!\$\ A_2^{\mathsf{Dec}^*(sk,\cdot)}(pk, c, St)$
If $d = b$ then return 1 else return 0

Oracle $\mathsf{Dec}^*(sk, c)$
$m \leftarrow \mathsf{Dec}(sk, c)$
$r' \leftarrow \mathsf{Rec}(sk, c)$
Return (m, r')

Above we require that the output of A_1 satisfies $|m_0| = |m_1|$ and that A_2 does not query c to its oracle. Define the *ind-ecca advantage* of A against PKE as

$$\mathbf{Adv}^{\text{ind-ecca}}_{\mathsf{PKE},A}(k) = 2 \cdot \Pr\left[\mathbf{Exp}^{\text{ind-ecca}}_{\mathsf{PKE},A}(k) \text{ outputs } 1\right] - 1\ .$$

We say that PKE is *enhanced chosen-ciphertext secure* (ECCA-secure) if $\mathbf{Adv}^{\text{ind-ecca}}_{\mathsf{PKE},A}(\cdot)$ is negligible for every efficient A.

Note that when PKE is randomness recovering, the ECCA experiment is efficient. In general, however, one can still ask whether a scheme meets the notion of ECCA even when it is not randomness recovering. In this case, it may still be possible to simulate the ECCA experiment efficiently since in the proof of security we are additionally given the code of the adversary A (and so, for example, the randomness for encryption might be efficiently extractable from the code of A using non-black-box techniques). We leave exploration of ECCA security relative to an inefficient Rec algorithm for future work.

(Not) allowing decryption error. Unless otherwise specified, we will always require that an ECCA-secure PKE scheme has *perfect correctness*. Indeed, curiously, it turns out that an ECCA-secure, randomness-recovering PKE scheme is easy to construct given any CCA-secure one if we allow *negligible decryption error* — however, an ECCA-secure scheme with negligible decryption error will not be sufficient in the applications we consider.[5] (This observation and example are due to [1].)

Let $\mathsf{PKE} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ be a CCA-secure scheme. Assume messages and the coins used by the encryption algorithm for this scheme both have length k, and let $n(k)$ be the length of a ciphertext produced by Enc. We construct a scheme $\mathsf{PKE}' = (\mathsf{Kg}, \mathsf{Enc}', \mathsf{Dec}, \mathsf{Rec}')$ in which the coins used by the encryption algorithm have length $2k + n(k)$, defined via:

Alg $\mathsf{Enc}'(pk, m; r')$
$r_1 \| r_2 \| r_3 \leftarrow r'$ // $|r_1| = |r_2| = k, |r_3| = n(k)$
If $r_1 = 0^k$ then return r_3
Else return $\mathsf{Enc}(pk, m; r_2)$

Alg $\mathsf{Rec}'(sk, c)$
Return $0^k \| 0^k \| c$

Proposition 3.1 *Suppose* PKE *is CCA-secure. Then* PKE' *is randomness-recovering and ECCA-secure. However,* PKE' *has decryption error* 2^{-k}.

Proof: The decryption error is evident from the construction. We next claim that PKE' is randomness-recovering. This is so because if $c = \mathsf{Enc}(pk, m; r)$ then $\mathsf{Enc}'(pk, m; 0^k || 0^k || c) = c$. We also claim that PKE' is ECCA-secure. The intuition is that oracle $\mathsf{Dec}^*(sk, c)$ in the ECCA game returns $m = \mathsf{Dec}(sk, c)$ as per the original scheme and also returns $0^k || 0^k || c$, but the latter is not additional information for the adversary since it already had c (which it queried). We omit the formal proof. ∎

CCA does not imply ECCA. A next natural question to ask is whether, assuming perfect correctness, ECCA security is stronger requirement than CCA security. We answer this question affirmatively by showing that, given a perfectly correct, CCA-secure randomness-recovering PKE scheme, we can construct another randomness-recovering PKE scheme that is still CCA-secure but is *not* ECCA-secure. This motivates the construction of specialized ECCA-secure schemes in Section 4.

[5] The resulting ECCA-secure scheme does not have *unique* randomness recovery, though. In the case of unique randomness recovery, schemes with negligible decryption error may still have some applications, but for simplicity we do not discuss it in the paper.

Consider a randomness-recovering CCA-secure scheme $\mathsf{PKE} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$. We transform PKE to a new scheme $\mathsf{PKE}^* = (\mathsf{Kg}^*, \mathsf{Enc}^*, \mathsf{Dec}^*)$ which is still CCA-secure but is *not* ECCA-secure. The idea is to embed a "test" ciphertext in the public key of the new scheme, such that its decryption algorithm returns the secret key if given as input some randomness consistent with this test ciphertext. Formally, PKE^* is constructed as follows (where we implicitly assume the public key is contained in the secret key):

$\mathbf{Alg}\ \mathsf{Kg}^*(1^k)$	$\mathbf{Alg}\ \mathsf{Enc}^*((pk, c^*), m)$	$\mathbf{Alg}\ \mathsf{Dec}^*(sk, c\|\|b)$
$(pk, sk) \leftarrow\!\!\$\ \mathsf{Kg}(1^k)$	$c \leftarrow\!\!\$\ \mathsf{Enc}(pk, m)$	If $b = 1$ and $\mathsf{Enc}(pk, 0; c) = c^*$
$r \leftarrow\!\!\$\ \{0,1\}^k$	Return $c\|\|0$	then return sk
$c^* \leftarrow \mathsf{Enc}(pk, 0\,; r)$		Return $\mathsf{Dec}(sk, c)$
Return $((pk, c^*), sk)$		

Note that using the extra "flag bit" appended to ciphertexts ensures that PKE^* maintains perfect correctness.

Proposition 3.2 *Assuming* PKE *is CCA-secure and has perfect correctness,* PKE^* *is CCA-secure but is not ECCA-secure.*

Proof: To show that PKE^* is not ECCA-secure, consider the following ECCA adversary $A = (A_1, A_2)$ against it:

- Algorithm A_1 on input the public key (pk, c^*) queries c^* to the decryption oracle to receive $(0, r')$ for some r'. It then queries $r'\|1$ to the decryption oracle and receives sk. Finally, it outputs $(0, 1)$ as the challenge messages and sk as the state.
- A_2 on inputs $(pk, c^*), c\|0, sk$ computes $b \leftarrow \mathsf{Dec}(sk, c)$ and returns b.

It is clear that A achieves ECCA advantage 1.

We now need to show that PKE^* remains CCA-secure. We sketch a hybrid argument to show this. Let A be a CCA adversary against PKE^*. We define a second game where we replace the public key in the CCA experiment (pk, c^*) with c^* generated as $c^* \leftarrow \mathsf{Enc}(pk, 1; r)$. If A's advantage differs significantly between these two games, we can contradict CCA security of PKE via the following CCA adversary $A' = (A'_1, A'_2)$:

- Algorithm A'_1 on input pk outputs $(0, 1)$ as the challenge messages.
- On inputs pk, c^*, algorithm A'_2 runs A_1 on input (pk, c^*), answering any decryption query c as follows: if $c = c^*$ then return 0, else return $\mathsf{Dec}(sk, c)$ using its own decryption oracle. Eventually A_1 outputs (m_0, m_1, St). Then A'_2 picks a random bit b and sets $c \leftarrow \mathsf{Enc}(pk, m_b)$. It runs A_2 on inputs $(pk, c^*), c, St$, answering decryption queries as before.

Note that in this second game, A's decryption oracle for Dec^* behaves identically to a decryption oracle for Dec, assuming PKE has perfect correctness: Indeed, the only way A's decryption oracle could behave differently from Dec is for A to query some $r'\|1$ such that $\mathsf{Enc}(pk, 0; r') = c^*$. But because of how the game works we know that also $\mathsf{Enc}(pk, 1; r) = c^*$ for some r. But $\mathsf{Dec}(sk, c^*)$ is a single value, either 0 or 1, and thus violating perfect correctness. ∎

TAG-BASED DEFINITION. Let $\mathsf{TB\text{-}PKE} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ be a tag-based public-key encryption scheme with tag-space $TagSp$. We associate to $\mathsf{TB\text{-}PKE}$ and an adversary $A = (A_1, A_2, A_3)$ a *tag-based enhanced chosen-ciphertext attack* experiment,

$\mathbf{Experiment}\ \mathbf{Exp}^{\text{ind-tb-ecca}}_{\mathsf{TB\text{-}PKE},A}(k)$	$\mathbf{Oracle}\ \mathsf{Dec}^*(sk, t, c)$
$b \leftarrow\!\!\$\ \{0,1\}\ ;\ (pk, sk) \leftarrow\!\!\$\ \mathsf{Kg}(1^k)$	$m \leftarrow \mathsf{Dec}(sk, t, c)$
$t \leftarrow\!\!\$\ A_1(1^k)$	$r' \leftarrow \mathsf{Rec}(sk, t, c)$
$(m_0, m_1, St) \leftarrow\!\!\$\ A_2^{\mathsf{Dec}^*(sk,\cdot,\cdot)}(pk, t)$	Return (m, r')
$c \leftarrow\!\!\$\ \mathsf{Enc}(pk, t, m_b)$	
$d \leftarrow\!\!\$\ A_3^{\mathsf{Dec}^*(sk,\cdot,\cdot)}(pk, t, c, St)$	
If $d = b$ then return 1 else return 0	

Above we require that the output of A_2 satisfies $|m_0| = |m_1|$ and that A_3 does not make a query of the form $\mathsf{Dec}^*(sk, t, \cdot)$ to its oracle. Define the *ind-tb-ecca advantage* of A against PKE as

$$\mathbf{Adv}^{\text{ind-tb-ecca}}_{\mathsf{PKE},A}(k) = 2 \cdot \Pr\left[\, \mathbf{Exp}^{\text{ind-tb-ecca}}_{\mathsf{PKE},A}(k) \text{ outputs } 1 \,\right] - 1 \,.$$

We say that $\mathsf{TB\text{-}PKE}$ is *tag-based enhanced chosen-ciphertext secure* (TB-ECCA-secure) if $\mathbf{Adv}^{\textit{ind-tb-ecca}}_{\mathsf{PKE},A}(\cdot)$ is negligible for every efficient A.

4 Constructions of ECCA-Secure PKE

We now detail several constructions of ECCA secure encryption. They are based on notions of adaptivity for trapdoor functions introduced in [29] so accordingly we recall those first.

4.1 Adaptivity for Trapdoor Functions

Adaptive trapdoor functions. Let $\mathsf{TDF} = (\mathsf{Tdg}, \mathsf{Eval}, \mathsf{Inv})$ be an (injective) trapdoor function family. We associate to TDF and an inverter I an *adaptive one-way* experiment,

$$\begin{array}{l}\textbf{Experiment } \mathbf{Exp}^{\text{aow}}_{\mathsf{TDF},I}(k) \\ \quad (ek, td) \leftarrow\!\!\$\ \mathsf{Tdg}(1^k)\ ;\ x \leftarrow\!\!\$\ \{0,1\}^k \\ \quad y \leftarrow \mathsf{Eval}(ek, x) \\ \quad x' \leftarrow\!\!\$\ I^{\mathsf{Inv}(td,\cdot)}(ek, y) \\ \quad \text{If } x = x' \text{ then return 1 else return 0}\end{array}$$

Above we require that I does not query y to its oracle. Define the *aow-advantage* of A against TDF as

$$\mathbf{Adv}^{\text{aow}}_{\mathsf{TDF},I}(k) = \Pr\left[\, \mathbf{Exp}^{\text{aow}}_{\mathsf{TDF},I}(k) \text{ outputs } 1 \,\right] .$$

We say that TDF is *adaptive one-way* (or is an ATDF) if $\mathbf{Adv}^{\text{aow}}_{\mathsf{TDF},I}(\cdot)$ is negligible for every efficient I.

Tag-based adaptivity. Let $\mathsf{TB\text{-}TDF} = (\mathsf{Tdg}, \mathsf{Eval}, \mathsf{Inv})$ be a tag-based trapdoor function family. We associate to TDF and an inverter $I = (I_1, I_2)$ a *tag-based adaptive one-way* experiment,

$$\begin{array}{l}\textbf{Experiment } \mathbf{Exp}^{\text{tb-aow}}_{\mathsf{TB\text{-}TDF},I}(k) \\ \quad (ek, td) \leftarrow\!\!\$\ \mathsf{Tdg}(1^k) \\ \quad t \leftarrow\!\!\$\ I_1(1^k)\ ;\ x \leftarrow\!\!\$\ \{0,1\}^k \\ \quad y \leftarrow \mathsf{Eval}(ek, t, x) \\ \quad x' \leftarrow\!\!\$\ I_2^{\mathsf{Inv}(td,\cdot,\cdot)}(ek, t, y) \\ \quad \text{If } x = x' \text{ then return 1 else return 0}\end{array}$$

Above we require that I_2 does not make a query of the form $\mathsf{Inv}(td, t, \cdot)$ to its oracle. Define the *tb-aow-advantage* of A against $\mathsf{TB\text{-}TDF}$ as

$$\mathbf{Adv}^{\text{tb-aow}}_{\mathsf{TB\text{-}TDF},I}(k) = \Pr\left[\, \mathbf{Exp}^{\text{tb-aow}}_{\mathsf{TB\text{-}TDF},I}(k) \text{ outputs } 1 \,\right] .$$

We say that TDF is *tag-based adaptive one-way* (or is a TB-ATDF) if $\mathbf{Adv}^{\text{tb-aow}}_{\mathsf{TDF},I}(\cdot)$ is negligible for every efficient I.

Realizations. In [29] it is shown that ATDFs and tag-based ATDFs can be realized from lossy TDFs [36] and correlated-product secure TDFs [38], which can be realized from a variety of standard number-theoretic and lattice-based assumptions. Furthermore, tag-based ATDFs were constructed from a strong but non-decisional (*i.e., search*) problem on RSA in [29].

4.2 ECCA Security from Adaptive Trapdoor Functions

Here we construct ECCA-secure public-key encryption from adaptive TDFs. We note that our construction applies to general ATDFs; in the case of ATDFs with a linear number of hardcore bits we obtain a much more efficient construction, see Appendix B for details.

OVERVIEW AND INTUITION. As in [29] (which constructs CCA-secure PKE from ATDFs), our approach involves first constructing a one-bit encryption scheme and then transforming it into a multi-bit scheme. In doing so we heavily use the recent approach of Hohenberger *et al.* [27] and their notion of *detectable* CCA security (DCCA); this should be contrasted with [29] who rely on [33] instead. Let us explain why.

Both [27] and [33] provide a way to "tie together" many one-bit ciphertexts via "inner" and "outer" encryption layers but differ in which layer contains the one-bit ciphertets. In [33], the inner layer is a multi-bit q-bounded non-malleable encryption scheme while the outer later is the concatenation of one-bit ciphertexts. This means that without a randomness-recovering inner layer, [33] does *not* preserve randomness-recovery of the outer one-bit scheme. Such an inner layer seems hard to construct, as known approaches to non-malleability [35, 12] crucially use randomness in an un-invertible way in their encryption algorithms (e.g., to generate a signature key-pair or a zero-knowledge proof).

On the other hand, in Hohenberger *et al.* [27] it is the inner layer that is the concatenation of one-bit ciphertexts, which obviates the problem since this inner layer is also used to encrypt randomness for use by the outer layer and thus the latter does *not* need to be randomness-recovering for the overall scheme to be so. Surprisingly, we also show that when this inner layer is randomness recovering then in all hybrid games used for the security proof the simulator is even able to the return randomness corresponding to valid ciphertexts, and thus the overall scheme also has ECCA security.

4.2.1 Enhanced DCCA Security

The notion of Detectable Chosen Ciphertext (DCCA) security was recently introduced by [27]. We define here the notion of *enhanced* DCCA (EDCCA) security, which parallels the notion of enhanced CCA security. In our definition, we require that the DCCA scheme be both enhanced and randomness-recovering. This is due to the fact that our application of DCCA requires both properties. However, the more general notion of enhanced DCCA security (with no efficient randomness-recovering property) may also be of interest.

DETECTABLE ENCRYPTION SCHEMES. A *detectable encryption scheme* is a tuple of probabilistic polynomial time algorithms $(\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec}, \mathcal{F})$ such that: (1) $(\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ constitute a public-key encryption scheme, and (2) $\mathcal{F}\colon (pk, c', c) \mapsto b \in \{0,1\}$, the detecting function $\mathcal{F}$ takes as input a public key pk and two ciphertexts c', c and outputs a bit.

Additionally, the detecting function $\mathcal{F}$ must have the following property: Informally, given the description of $\mathcal{F}$ and a public key pk, it it should be hard to find a second ciphertext c' that is related to a "challenge" ciphertext c, i.e. such that $\mathcal{F}(pk, c', c) = 1$, *before being given* c. See [27] for the formal definition of the unpredictability experiment.

EDCCA DEFINITION. We are now ready to define enhanced, detectable chosen ciphertext security. Let $\mathsf{PKE} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Rec}, \mathcal{F})$ be a randomness-recovering public-key encryption scheme. We associate to PKE and an adversary $A = (A_1, A_2)$ an *enhanced detectable chosen-ciphertext attack* experiment,

Experiment $\mathbf{Exp}^{\text{ind-edcca}}_{\mathsf{PKE},A}(k)$
$b \leftarrow\!\!\$\ \{0,1\}$; $(pk, sk) \leftarrow\!\!\$\ \mathsf{Kg}(1^k)$
$(m_0, m_1, St) \leftarrow\!\!\$\ A_1^{\mathsf{Dec}^*(\perp, sk, \cdot)}(pk)$
$c^* \leftarrow\!\!\$\ \mathsf{Enc}(pk, m_b)$
$d \leftarrow\!\!\$\ A_2^{\mathsf{Dec}^*(c^*, sk, \cdot)}(pk, c^*, St)$
If $d = b$ then return 1 else return 0

Oracle $\mathsf{Dec}^*(c^*, sk, c)$
If $c^* \neq \perp$ and $\mathcal{F}(pk, c^*, c) = 1$
then return $\perp$
Else $m \leftarrow \mathsf{Dec}(sk, c)$
$r' \leftarrow \mathsf{Rec}(sk, c)$
Return (m, r')

Above we require that the output of A_1 satisfies $|m_0| = |m_1|$ and that A_2 does not query c^* to its oracle. Define the *ind-edcca advantage* of A against PKE as

$$\mathbf{Adv}^{\text{ind-edcca}}_{\mathsf{PKE},A}(k) = 2 \cdot \Pr\left[\mathbf{Exp}^{\text{ind-edcca}}_{\mathsf{PKE},A}(k) \text{ outputs } 1\right] - 1\ .$$

We say that PKE is *enhanced detectable chosen-ciphertext secure* (EDCCA-secure) if

- Encryptions are indistinguishable: $\mathbf{Adv}^{\text{ind-edcca}}_{\mathsf{PKE},A}(\cdot)$ is negligible for every efficient A, AND
- $\mathcal{F}$ is unpredictable: Every efficient adversary A has negligible probability of succeeding in the unpredictability experiment (see [27]).

4.2.2 EDCCA Security from ATDFs

We construct an EDCCA scheme from ATDFs as follows: Let $\mathsf{TDF} = (\mathsf{Tdg}, \mathsf{Eval}, \mathsf{Inv})$ be a trapdoor function with hardcore bit hc, for example the Goldreich-Levin bit [23]. Define the following multi-bit public-key encryption scheme $\mathsf{EDCCA}[\mathsf{TDF}] = (\mathsf{Kg}_\mathsf{D}, \mathsf{Enc}_\mathsf{D}, \mathsf{Dec}_\mathsf{D})$:

$\mathbf{Alg}\ \mathsf{Kg}_\mathsf{D}(1^k)$	$\mathbf{Alg}\ \mathsf{Enc}_\mathsf{D}(ek, m = m_1, \ldots, m_\ell)$	$\mathbf{Alg}\ \mathsf{Dec}(td, C)$
$(ek, td) \leftarrow\!\!\$\ \mathsf{Tdg}(1^k)$	$x_1 \leftarrow\!\!\$\ \{0,1\}^k; \ldots; x_\ell \leftarrow\!\!\$\ \{0,1\}^k$	Parse $C = (y_1, \beta_1, \ldots, y_\ell, \beta_\ell)$
Return (ek, td)	Return $C = (\mathsf{Eval}(ek, x_1), \mathsf{hc}(x_1) \oplus m_1,$	For $1 \le i \le \ell$
	$\ldots, \mathsf{Eval}(ek, x_\ell), \mathsf{hc}(x_\ell) \oplus m_\ell)$	$m_i = \mathsf{hc}(\mathsf{Inv}(td, y_i)) \oplus \beta_i$
		Return $m_1, \ldots, m_\ell$

The Detecting Function $\mathcal{F}_\mathsf{D}$: On input $pk, C^* = (y_1^*, \beta_1^*, \ldots, y_\ell^*, \beta_\ell^*)$ and $C = (y_1, \beta_1, \ldots, y_\ell, \beta_\ell)$ (where the β_i^*, β_i are bits), we define:

$$\mathcal{F}_\mathsf{D}(pk, C^*, C) = \begin{cases} 1 & \text{if for some } i, j \in [\ell], y_i^* = y_j \\ 0 & \text{otherwise} \end{cases}$$

Claim 4.1 Suppose TDF is adaptive one-way. Then $\mathsf{EDCCA}[\mathsf{TDF}]$ defined above is a multi-bit EDCCA-secure encryption scheme.

We give some intuition for why the claim holds. Assume towards contradiction that we have an efficient adversary $A = (A_1, A_2)$ breaking $\mathsf{EDCCA}[\mathsf{TDF}]$ by distinguishing encryptions of two messages m_0, m_1 with advantage $1/p(k)$ for some polynomial $p(\cdot)$. Using a standard hybrid argument, we have that there must be some index $1 \le i \le \ell$ such that A successfully distinguishes encryptions of the message $m_0^* = m_0^1, \ldots, m_0^{i-1}, m_0^i, m_1^{i+1}, \ldots, m_1^\ell$ from encryptions of the message $m_1^* = m_0^1, \ldots, m_0^{i-1}, m_1^i, m_1^{i+1}, \ldots, m_1^\ell$, where m_b^j denotes the j-th bit of message m_b, with advantage $1/(\ell \cdot p(k))$. Note that the two messages, m_0^*, m_1^*, differ only in the setting of the i-th bit. Now assuming the existence of A, we construct an efficient adversary A' breaking adaptive one-wayness of TDF.

More specifically, A' receives $\tilde{y} = \mathsf{Eval}(ek, \tilde{x})$ externally and uses A to guess $\mathsf{hc}(\mathsf{Inv}(td, \tilde{y}))$ with advantage $1/(\ell \cdot p(k))$. A' does this by simulating the EDCCA decryption oracle for $A = (A_1, A_2)$ using its inversion oracle Inv.

First, we consider simulating responses to queries made by A_1. In this case, we have that with overwhelming probability, for every decryption query $C = (y_1, \beta_1, \ldots, y_\ell, \beta_\ell)$ made by A_1, it is the case that $y_j \neq \tilde{y}$ for all $1 \le j \le \ell$. Thus, A' can decrypt correctly using oracle access to Inv.

At the end of the first phase, A' prepares the challenge ciphertext C^* by choosing $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots x_\ell$ and a bit $\tilde{\beta}$ uniformly at random and setting

$$C^* = \left(\mathsf{Eval}(ek, x_1), \mathsf{hc}(x_1) \oplus m_0^1, \ldots, \tilde{y}, \tilde{\beta}, \ldots, \mathsf{Eval}(ek, x_\ell), \mathsf{hc}(x_\ell) \oplus m_1^\ell\right).$$

Now, to answer EDCCA decryption queries C submitted by A_2, A' checks whether $\mathcal{F}_\mathsf{D}(pk, C^*, C) = 1$. If yes, A' perfectly simulates the decryption oracle by returning $\perp$. If not, then this implies in particular that on input $C = (y_1, \beta_1, \ldots, y_\ell, \beta_\ell)$, we have that $y_j \neq \tilde{y}$ for all $1 \le j \le \ell$. In this case, A' can use its access to the Inv oracle in order to respond correctly to the decryption query.

Finally, A returns a bit d. A' computes $\alpha = m_d^i \oplus \tilde{\beta}$ and guesses that $\mathsf{hc}(\mathsf{Inv}(td, \tilde{y})) = \alpha$, thus succeeding with advantage $1/(\ell \cdot p(k))$.

We omit the technical definition of A', the analysis of the success probability of A' and the reduction from guessing a hardcore bit with non-negligible advantage to inverting the adaptive trapdoor function with non-negligible probability, since they are standard.

Remark 4.2 Scheme $\mathsf{EDCCA}[\mathsf{TDF}]$ defined above is also uniquely randomness-recovering. (This will be crucial for our application to adaptive trapdoor functions in Section 5.) It is also perfectly correct.

4.2.3 From EDCCA to ECCA Security

We next show that the construction of [27] designed to build a CCA-secure scheme from a DCCA secure one allows us to go from EDCCA to ECCA. That is, beyond what was already shown in [27] we show that the construction preserves "enhanced" security; it also preserves (unique) randomness-recoverability. Specifically, we instantiate the construction of [27] with the above randomness recovering EDCCA scheme, a CPA-secure scheme with perfect correctness, and a 1-bounded CCA-secure[6] scheme with perfect correctness (note that all these components can be constructed in a black-box manner from ATDFs):

The EDCCA scheme, $\mathsf{EDCCA}[\mathsf{TDF}]$: We instantiate the EDCCA scheme with the scheme given in Section 4.2.2. We note that for simplicity, we sometimes refer to the detecting function $\mathcal{F}_{\mathsf{D}}$ as checking for a "quoting" attack on the challenge ciphertext.

The CPA scheme, $\mathsf{CPA}[\mathsf{TDF}]$: We instantiate the CPA scheme with the same scheme $\mathsf{EDCCA}[\mathsf{TDF}]$ as above. Note that this scheme has perfect correctness since the Inv algorithm of the ATDF is required to invert correctly with probability 1.

The 1-bounded CCA scheme, $\mathsf{1\text{-}CCA}[\mathsf{TDF}]$: Since we have already observed above that we can construct a multi-bit CPA scheme with perfect correctness from ATDFs, we may now use any construction of a multi-bit 1-bounded CCA scheme with perfect correctness from a multi-bit CPA scheme with perfect correctness. This can be done in a black-box manner via the [12] construction. It is not hard to see that the construction of [12] preserves the perfect correctness property.

The Multi-Bit (Uniquely Randomness Recovering) ECCA Scheme. We present a multi-bit, uniquely randomness recovering, ECCA-secure encryption scheme $\mathsf{PKE}[\mathsf{TDF}] = (\mathsf{Kg}_{\mathsf{ECCA}}, \mathsf{Enc}_{\mathsf{ECCA}}, \mathsf{Dec}_{\mathsf{ECCA}})$ using the schemes $\mathsf{EDCCA}[\mathsf{TDF}] = (\mathsf{Kg}_{\mathsf{D}}, \mathsf{Enc}_{\mathsf{D}}, \mathsf{Dec}_{\mathsf{D}})$, $\mathsf{1\text{-}CCA}[\mathsf{TDF}] = (\mathsf{Kg}_{\mathsf{1b}}, \mathsf{Enc}_{\mathsf{1b}}, \mathsf{Dec}_{\mathsf{1b}})$, and $\mathsf{CPA}[\mathsf{TDF}] = (\mathsf{Kg}_{\mathsf{CPA}}, \mathsf{Enc}_{\mathsf{CPA}}, \mathsf{Dec}_{\mathsf{CPA}})$ defined above.

Alg $\mathsf{Kg}_{\mathsf{ECCA}}(1^\lambda)$
$(pk_{in}, sk_{in}) \leftarrow\!\!\$\ \mathsf{Kg}_{\mathsf{D}}(1^\lambda)$
$(pk_A, sk_A) \leftarrow\!\!\$\ \mathsf{Kg}_{\mathsf{1b}}(1^\lambda)$
$(pk_B, sk_B) \leftarrow\!\!\$\ \mathsf{Kg}_{\mathsf{CPA}}(1^\lambda)$
$pk \leftarrow (pk_{in}, pk_A, pk_B)$
$sk \leftarrow (sk_{in}, sk_A, sk_B)$
Return (pk, sk)

Alg $\mathsf{Enc}_{\mathsf{ECCA}}(pk, m)$
$(r_A, r_B) \leftarrow\!\!\$\ \{0,1\}^\lambda$
$CT_{in} \leftarrow\!\!\$\ \mathsf{Enc}_{\mathsf{D}}(pk_{in}, (r_A, r_B, m))$
$CT_A \leftarrow \mathsf{Enc}_{\mathsf{1b}}(pk_A, CT_{in}; r_A)$
$CT_B \leftarrow \mathsf{Enc}_{\mathsf{CPA}}(pk_B, CT_{in}; r_B)$
Return $CT = (CT_A, CT_B)$

Alg $\mathsf{Dec}_{\mathsf{ECCA}}(sk, CT)$
$CT_{in} \leftarrow\!\!\$\ \mathsf{Dec}_{\mathsf{1b}}(sk_A, CT_A)$
$(r_A, r_B, m) \leftarrow \mathsf{Dec}_{\mathsf{D}}(sk_{in}, CT_{in})$
$r_{in} \leftarrow \mathsf{Rec}_{\mathsf{D}}(sk_{in}, CT_{in})$
If $CT_A = \mathsf{Enc}_{\mathsf{1b}}(pk_A, CT_{in}; r_A)$
and $CT_B = \mathsf{Enc}_{\mathsf{CPA}}(pk_B, CT_{in}; r_B)$
return (r_A, r_B, m, r_{in})
Else return $\perp$

Theorem 4.3 $\mathsf{PKE}[\mathsf{TDF}]$ *is enhanced CCA-secure and uniquely randomness recovering with perfect correctness under the assumptions that* $\mathsf{EDCCA}[\mathsf{TDF}]$ *is enhanced DCCA-secure and uniquely randomness recovering,* $\mathsf{1\text{-}CCA}[\mathsf{TDF}]$ *is 1-bounded CCA secure with perfect correctness, and* $\mathsf{CPA}[\mathsf{TDF}]$ *is CPA-secure with perfect correctness.*

[6] By 1-bounded CCA security, we mean an encryption scheme that is secure under an indistinguishability attack when the adversary may make only a single decryption query to its oracle either before or after receiving the challenge ciphertext.

Note that Theorem 4.3 implies that there is a black-box construction of multi-bit, uniquely randomness recovering, enhanced CCA-secure encryption from ATDF.

Proof: The proof is based on [27]. We begin by defining a game which is slightly different than the regular enhanced CCA game, but will be useful in our analsysis of $\mathsf{PKE}[\mathsf{TDF}]$:

Enhanced Nested Indistinguishability Game for scheme $\mathsf{PKE}[\mathsf{TDF}]$: We associate to the scheme $\mathsf{PKE}[\mathsf{TDF}]$ and to an adversary $A = (A_1, A_2)$ an *enhanced nested indistinguishability under chosen ciphertext attack* experiment,

$$
\begin{array}{ll}
\textbf{Experiment } \mathbf{Exp}^{\text{nested-ind-ecca}}_{\mathsf{PKE}[\mathsf{TDF}],A}(k) & \textbf{Oracle } \mathsf{Dec}^{*}_{\mathsf{ECCA}}(sk, c) \\
\quad b, z \leftarrow\!\!\$\ \{0,1\}\ ;\ (pk, sk) \leftarrow\!\!\$\ \mathsf{Kg}_{\mathsf{ECCA}}(1^k) & \quad m \leftarrow \mathsf{Dec}_{\mathsf{ECCA}}(sk, c) \\
\quad (m_0, m_1, St) \leftarrow\!\!\$\ A_1^{\mathsf{Dec}^{*}_{\mathsf{ECCA}}(sk,\cdot)}(pk) & \quad r' \leftarrow \mathsf{Rec}_{\mathsf{ECCA}}(sk, c) \\
\quad r_A, r_B \leftarrow\!\!\$\ \{0,1\}^{\lambda} & \quad \text{Return } (m, r') \\
\quad \text{If } z = 0 & \\
\quad\quad CT^{*}_{in} \leftarrow\!\!\$\ \mathsf{Enc}_{\mathsf{D}}(pk_{in}, (r_A, r_B, m_b)) & \\
\quad \text{Else if } z = 1 & \\
\quad\quad CT^{*}_{in} \leftarrow\!\!\$\ \mathsf{Enc}_{\mathsf{D}}(pk_{in}, 0^{|r_A|+|r_A|+|m_b|}) & \\
\quad CT^{*}_{A} \leftarrow \mathsf{Enc}_{\mathsf{1b}}(pk_A, CT^{*}_{in}; r_A) & \\
\quad CT^{*}_{B} \leftarrow \mathsf{Enc}_{\mathsf{CPA}}(pk_B, CT^{*}_{in}; r_B) & \\
\quad CT^{*} \leftarrow (CT^{*}_{A}, CT^{*}_{B}) & \\
\quad z' \leftarrow\!\!\$\ A_2^{\mathsf{Dec}^{*}_{\mathsf{ECCA}}(sk,\cdot)}(pk, CT^{*}, St) & \\
\quad \text{If } z' = z \text{ then return 1 else return 0} &
\end{array}
$$

Above we require that the output of A_1 satisfies $|m_0| = |m_1|$ and that A_2 does not query CT^* to its oracle. In the following, we refer to decryption queries made by A_1 as "Phase 1 queries" and to decryption queries made by A_2 as "Phase 2 queries."

Define the *nested-ind-ecca advantage* of A against $\mathsf{PKE}[\mathsf{TDF}]$ as

$$\mathbf{Adv}^{\text{nested-ind-ecca}}_{\mathsf{PKE}[\mathsf{TDF}],A}(k) = 2 \cdot \Pr\left[\mathbf{Exp}^{\text{nested-ind-ecca}}_{\mathsf{PKE}[\mathsf{TDF}],A}(k) \text{ outputs } 1\right] - 1\ .$$

We say that $\mathsf{PKE}[\mathsf{TDF}]$ has *enhanced nested indistinguishable encryptions under a chosen ciphertext attack* if $\mathbf{Adv}^{\text{nested-ind-ecca}}_{\mathsf{PKE}[\mathsf{TDF}],A}(\cdot)$ is negligible for every efficient A.

It should be clear that enhanced nested indistinguishability of $\mathsf{PKE}[\mathsf{TDF}]$ under a chosen ciphertext attack implies enhanced CCA security of $\mathsf{PKE}[\mathsf{TDF}]$ (via a simple hybrid argument).

Consider the following event:

Definition 4.4 (The Bad Query Event) *We say that a* bad query event *has occurred during an execution of this experiment if in Phase 2, the adversary A makes a decryption query of the form $CT = (CT_A, CT_B)$ such that*

- *(Quoting attack on inner ciphertext:) $\mathcal{F}_{\mathsf{D}}(pk_{in}, CT^{*}_{in}, \mathsf{Dec}_{\mathsf{1b}}(sk_A, CT_A)) = 1$ AND*
- *(Query ciphertext differs from challenge ciphertext in first half:) $CT^{*}_{A} \neq CT_A$.*

We will show that Bad Query Event occurs with at most negligible probability when $z = 1$ and when $z = 0$. Once we have shown this, Nested Indistinguishability of $\mathsf{PKE}[\mathsf{TDF}]$ will follow in a straightforward manner.

Lemma 4.5 *Bad Query Event occurs with negligible probability when $z = 1$.*

We prove Lemma 4.5 via a sequence of hybrids:

Hybrid H_0: Proceeds exactly as the nested indistinguishability game for the case where $z = 1$.

Hybrid H_1: Proceeds exactly like H_0 except that CT^*_B is set to be: $CT^*_B = \mathsf{Enc}_{\mathsf{CPA}}(pk_B, 1^k; r_B)$.

Claim 4.6 The probability of a Bad Query Event in H_1 and H_0 differs by a negligible amount.

Since sk_B is never used by the decryption oracle and since the decryption oracle can detect all Bad Query Events, the claim follows immediately by a reduction to the semantic security of $\mathsf{CPA}[\mathsf{TDF}]$.

Hybrid H_2: Proceeds exactly like H_1 except CT^*_A is set to be $CT^*_A = \mathsf{Enc}_{\mathsf{1b}}(pk_A, 1^k; r_A)$.

Claim 4.7 The probability of Bad Query Event in H_2 is negligible.

The claim follows due to the fact that the challenge ciphertext in H_2 contains no information about CT^*_{in} and since the detecting function $\mathcal{F}_{\mathsf{D}}$ is unpredictable.

Claim 4.8 The probability of a Bad Query Event in H_2 and H_1 differs by a negligible amount.

Intuitively, Claim 4.8 will reduce to the 1-bounded CCA security of $\mathsf{1\text{-}CCA}[\mathsf{TDF}]$.

Proof: Assume towards contradiction that there is some efficient adversary A which causes Bad Query Event to occur with negligible probability in H_2 and non-negligible probability in H_1. We denote by q the (polynomial) number of Phase 2 queries made by A. By standard hybrid argument, there must be some index $i \in [q]$ such that the i-th Phase 2 query made by A in H_1 causes Bad Query Event to occur with non-negligible probability. On the other hand, for every index i, the i-th Phase 2 query made by A in H_2 causes Bad Query Event to occur with at most negligible probability.

Fix such i. We construct an adversary B which breaks the security of $\mathsf{1\text{-}CCA}[\mathsf{TDF}]$. B will receive a challenge ciphertext that is either an encryption of CT^*_{in} or of 1^n. In case the challenge ciphertext was an encryption of CT^*_{in}, B will perfectly simulate the adversary's view in H_1. In case the challenge ciphertext was an encryption of 1^n, B will perfectly simulate the adversary's view in H_2.

Moreover, B will be able to detect whether Bad Query Event occurred in the i-th Phase 2 query of the experiment. Thus, if Bad Query Event occurs in the i-th query with non-negligible probability in H_1 and negligible probability in H_2, then B will be able to break security of $\mathsf{1\text{-}CCA}[\mathsf{TDF}]$.

Formally, consider the following Simulated Decryption Oracle:

Simulated Decryption Oracle:

- Decrypt CT_B using sk_B to retreive CT_{in}.
- Decrypt CT_{in} using sk_{in} to retrieve (r_A, r_B, m).
- Use the randomness recovering algorithm $\mathsf{Rec}_{\mathsf{D}}$ and sk_{in} to retrieve $r_{in} = \mathsf{Rec}_D(sk_{in}, CT_{in})$.
- Check that CT_A and CT_B were formed correctly with respect to (r_A, r_B, CT_{in}). If not, output $\perp$. Otherwise, output (m, r_A, r_B, r_{in}).

Note that for every possible string CT submitted to the oracle, the output of the Simulated Decryption Oracle and the real decryption oracle is identical since 1-CCA[TDF] and CPA[TDF] have perfect correctness.

Now, B does the following: B receives pk_A from its external 1-bounded CCA challenger and generates $(sk_B, pk_B), (sk_{in}, pk_{in})$ honestly.

B and A interact in Phase 1 (while B uses the Simulated Decryption Oracle to respond to decryption queries). At some point A outputs m_0 and m_1. B computes $CT^*_{in} = \mathsf{Enc}_{\mathsf{D}}(pk_{in}, 0^\ell)$ and $CT^*_B = \mathsf{Enc}_{\mathsf{CPA}}(pk_B, 1^k)$. It outputs CT^*_{in} and 1^k as messages m_0, m_1 to its external 1-bounded CCA challenger and receives a ciphertext CT^*_A. B then outputs $CT^* = (CT^*_A, CT^*_B)$ to A. B continues interacting with A during Phase 2, while using the Simulated Decryption Oracle as above. When B receives the i-th Phase 2 query of A, denoted by (CT^i_A, CT^i_B), B checks for the Bad Query Event by doing the following:

- B checks if $CT^i_A \neq CT^*_A$.
- If so, B submits CT^i_A to its external 1-bounded CCA decryption oracle and receives CT^i_{in} in response.
- B checks whether $\mathcal{F}_{\mathsf{D}}(pk_{in}, CT^*_{in}, CT^i_{in}) = 1$ (i.e. whether a quoting attack occurred). If yes, B outputs 0. Otherwise, B outputs 1.

Since by assumption we have that Bad Query Event occurs with non-negligible probability at the i-th Phase 2 query in H_1 and occurs with negligible probability at the i-th Phase 2 query in H_2, we have that B achieves non-negligible advantage in the external 1-bounded CCA security game. This is a contradiction to the security of 1-CCA[TDF] and so the claim is proved. ▌

Lemma 4.5 follows immediately from Claims 4.6, 4.7 and 4.8.

We now turn to the case where $z = 0$:

Lemma 4.9 *Bad Query Event occurs with negligible probability when $z = 0$.*

Intuitively, Lemma 4.9 will reduce to the enhanced detectable CCA security of EDCCA[TDF].

Proof: We have already shown that when $z = 1$, Bad Query Event occurs with negligible probability. We will now show that if there is an efficient adversary A causing Bad Query Event to occur with non-negligible probability when $z = 0$, then there is a ppt adversary B breaking the security of EDCCA[TDF].

Assume towards contradiction that there is an efficient adversary A which causes Bad Query Event to occur with non-negligible probability when $z = 0$. Consider the following efficient adversary B which interacts with A in a run of the nested indistinguishability experiment, while externally participating in an enhanced DCCA indistinguishability experiment. B does the following:

Setup: B receives pk_{in} externally from the EDCCA experiment. B honestly generates $(sk_A, pk_A), (sk_B, pk_B)$.

Phase 1: B simulates the honest (enhanced) decryption oracle using sk_A and the decryption oracle in the external enhanced DCCA experiment. At the end of this phase A will output m_0, m_1.

Challenge: Choose random $\beta \in \{0,1\}$ and $r_A, r_B \in \{0,1\}^\lambda$. Send to the external enhanced DCCA challenger $M_0 = (r_A, r_B, m_\beta)$ and $M_1 = 0^{|M_0|}$ and obtain the ciphertext CT^*_{in}. Compute CT^*_A and CT^*_B honestly, given CT^*_{in}, r_A, r_B. Return $CT^* = (CT^*_A, CT^*_B)$ to A.

Phase 2: When A queries the decryption oracle on $CT = (CT_A, CT_B)$, compute $CT_{in} = \mathsf{Dec}_{\mathsf{1b}}(sk_A, CT_A)$.

Case 1 (a bad query event): $CT_A \neq CT_A^*$ and yet $\mathcal{F}_{\mathsf{D}}(pk_{in}, CT_{in}^*, CT_{in}) = 1$ (i.e. a quoting attack occurred), then abort and output the bit 0.

Case 2 (partial match with challenge): $CT_A = CT_A^*$, then return $\perp$ to A.

Otherwise, query the external EDCCA decryption oracle to decrypt CT_{in} and return its randomness. Check that CT_A and CT_B are consistent with the response. If not, return $\perp$. Otherwise, return the message and all randomness.

Output: When A outputs a bit, B outputs 0 or 1 with probability 1/2.

We argue that B correctly answers all decryption queries except when it aborts. This will follow immediately once we establish that B always answers correctly by returning $\perp$ when a Case 2 query occurs. We next show that this is indeed the case.

Since a decryption query on the challenge is forbidden by the experiment, if $CT_A = CT_A^*$, then $CT_B \neq CT_B^*$. However, in this case CT must be an invalid ciphertext. We see this as follows: Since decryption is deterministic, we have that $CT_{in} = \mathsf{Dec}_{\mathsf{1b}}(sk_A, CT_A) = \mathsf{Dec}_{\mathsf{1b}}(sk_A, CT_A^*)$ and $(r_A, r_B, m) = \mathsf{Dec}_{\mathsf{D}}(sk_{in}, CT_{in})$. But this means that there is only one possible ciphertext CT_B that matches $CT_A = CT_A^*$. Since the challenge CT^* is a valid ciphertext, CT_B^* must be this value and so $CT = (CT_A^*, CT_B)$ must be invalid.

Now, if a Case 1 query occurs, B cannot decrypt using its EDCCA decryption oracle and must abort the experiment. But in this case, B can already guess that $z = 0$ since when $z = 1$, Case 1 occurs with negligible probability.

More specifically, when B aborts, it causes the external EDCCA experiment to output 1 with high probability. Moreover, when B does not abort, it causes the external EDCCA experiment to output 1 with probability 1/2. Since B aborts with non-negligible probability when $z = 0$, B causes the experiment's output to be 1 with probability non-negligibly greater than 1/2. This is a contradiction to the security of $\mathsf{EDCCA}[\mathsf{TDF}]$ and so the claim is proved. ▌

Finally, assuming that Bad Query Event occurs with negligible probability both when $z = 0$ and $z = 1$, we show that Nested Indistinguishability under chosen ciphertext attacks holds. This is straightforward via a reduction to the enhanced DCCA security of $\mathsf{EDCCA}[\mathsf{TDF}]$. ▌

4.3 ECCA Security from Tag-Based Adaptive Trapdoor Functions

We next give more efficient constructions of ECCA-secure public-key encryption from *tag-based* adaptive trapdoor functions introduced by Kiltz *et al.* [29].

From tag-based ATDF to tag-based ECCA-secure PKE. It is straightforward to construct a multi-bit tag-based ECCA-secure PKE scheme from a tag-based ATDF, as follows. Let $\mathsf{TB\text{-}TDF} = (\mathsf{Tdg}, \mathsf{Eval}, \mathsf{Inv})$ be a tag-based adaptive trapdoor function family with tag space $TagSp$ and hardcore bit hc. Define the following tag-based public-key encryption scheme $\mathsf{TB\text{-}PKE}[\mathsf{TB\text{-}TDF}] = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ with tag space $TagSp$ and message space $\{0,1\}^\ell$:

Alg $\mathsf{Kg}(1^k)$
  $(ek, td) \leftarrow\!\!\$\ \mathsf{Tdg}(1^k)$
  Return (ek, td)

Alg $\mathsf{Enc}(ek, t, m)$
  For $i = 1$ to ℓ do:
    $x_i \leftarrow\!\!\$\ \{0,1\}^k$
    $c_{i,1} \leftarrow \mathsf{Eval}(ek, t, x_i)$
    $c_{i,2} \leftarrow \mathsf{hc}(x) \oplus m[i]$
  $\mathbf{c} \leftarrow ((c_{1,1}, c_{1,2}), \ldots, (c_{\ell,1}, c_{\ell,2}))$
  Return $\mathbf{c}$

Alg $\mathsf{Dec}(td, t, \mathbf{c})$
  $((c_{1,1}, c_{1,2}), \ldots, (c_{\ell,1}, c_{\ell,2})) \leftarrow \mathbf{c}$
  For $i = 1$ to ℓ do:
    $x_i \leftarrow \mathsf{Inv}(td, c_{i,1})$
    $m[i] \leftarrow \mathsf{hc}(x_i) \oplus c_{i,2}$
  Return m

Remark 4.10 Scheme TB-PKE[TB-TDF] defined above is uniquely randomness recovering.

Proposition 4.11 *Suppose* TB-TDF *is adaptive one-way. Then* TB-PKE[TB-TDF] *defined above is ECCA-secure.*

We omit the proof, which is routine.

From ECCA-secure tag-based PKE to ECCA-secure PKE. Note that Kiltz *et al.* [29] show a construction of CCA-secure PKE from any CCA-secure tag-based PKE using a strongly one-time unforgeable signature scheme. However, this construction does not preserve the randomness-recovering property or the ECCA security of the tag-based PKE. To get around this issue, and to construct ECCA-secure PKE from ECCA-secure tag-based PKE we employ a transformation of Boneh *et al.* [8], instead. Let $\mathsf{TB\text{-}PKE} = \mathsf{Kg}_{tag}, \mathsf{Enc}_{tag}, \mathsf{Dec}_{tag})$ be a tag-based public-key encryption scheme, H, g be hash functions, and $\mathsf{MAC} = (\mathsf{mac}, \mathsf{ver})$ be a message-authentication code. Define $\mathsf{PKE}[\mathsf{TB\text{-}PKE}, H, g, \mathsf{MAC}] = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$:

Alg $\mathsf{Kg}(1^k)$	**Alg** $\mathsf{Enc}(pk, m)$	**Alg** $\mathsf{Dec}(sk, (c_1, c_2, c_3))$
$(pk, sk) \leftarrow\!\!\$\ \mathsf{Kg}_{tag}(1^k)$	$x \leftarrow\!\!\$\ \{0,1\}^k$; $c_1 \leftarrow H(x)$	$m \| x \leftarrow \mathsf{Dec}_{tag}(sk, c_1, c_2)$
Return (pk, sk)	$c_2 \leftarrow\!\!\$\ \mathsf{Enc}_{tag}(pk, c_1, m \| x)$	If $H(x) \neq c_1$ then return $\perp$
	$c_3 \leftarrow \mathsf{mac}(g(x), c_2)$	If $\mathsf{ver}(g(x), c_2) = 1$ then return m
	Return (c_1, c_2, c_3)	Else return $\perp$

Remark 4.12 If TB-PKE is (uniquely) randomness recovering, so is $\mathsf{PKE}[\mathsf{TB\text{-}PKE}, H, g, \mathsf{MAC}]$. In particular, this is the case when $\mathsf{TB\text{-}PKE} = \mathsf{TB\text{-}PKE}[\mathsf{TB\text{-}TDF}]$ as defined above. Thus, we obtain a uniquely randomness-recovering ECCA-secure PKE scheme from any tag-based ATDF.

Proposition 4.13 *Suppose* TB-PKE *is ECCA-secure, H is target collision-resistant, g is pairwise-independent, and* MAC *is strongly unforgeable. Then* $\mathsf{PKE}[\mathsf{TB\text{-}PKE}, H, g, \mathsf{MAC}]$ *is ECCA-secure.*

Proof: We prove security using a sequence of hybrids. Our proof follows that of [8], and uses a deferred analysis technique originating from [22].

Hybrid H_0: The first game is the *ind-cca* game for $\mathsf{PKE}[\mathsf{TB\text{-}PKE}, H, g, \mathsf{MAC}]$ as defined earlier. Let $c^* = (c_1^*, c_2^*, c_3^*)$ be the challenge ciphertext, and x^* be random input used as input to the H when computing the challenge ciphertext.

Hybrid H_1: H_1 is the same as H_0 except that on decryption queries of the form (c_1^*, c_2, c_3) we always return $\perp$. Let *valid* be the event that this ciphertext is indeed valid (it has a valid decryption).

Obviously we have that $|\mathbf{Adv}_A^{H_1}(k) - \mathbf{Adv}_A^{H_0}(k)| \leq \Pr_1[valid]$. Let *coll* be the event that c_2 is valid and correctly decrypts to a message $m||x$ and at the same time we have $g(x) \neq g(x^*)$. Furthermore, let $forge$ be the event that $c_3 \leftarrow \mathsf{mac}(g(x^*), c_2)$. It is easy to see that $\Pr_1[valid] < \Pr_1[coll] + \Pr_1[forge]$. Note that $\Pr_1[coll]$ is negligible given the collision resistance of H. In particular, $g(x) \neq g(x^*)$ implies that $x \neq x^*$, which makes the pair (x, x^*) a collision for H. We note that this argument works in presence of an ECCA oracle as well. In particular, note that decryption queries of the form (c_1, c_2, c_3) are only answered if $c_1 \neq c_1^*$, and in this case, one can query c_2 to the decryption oracle for the tag-based PKE and recover both the underlying message and all the randomness used in generating the ciphertext (note that c_3 is deterministic given c_1 and c_2). Analyzing he bound on $\Pr_1[forge]$ is deferred to a later hybrid.

Hybrid H_2: H_2 is the same as H_1 except that when computing the challenge ciphertext we compute $c_2^* = \mathsf{Enc}_{tag}(pk, c_1, 0^{|m_0|} \| 0^k)$.

Note that $\mathbf{Adv}_A^{H_2}(k) = 1/2$ for any PPT adversary A. It is also straightforward to see that $|\mathbf{Adv}_A^{H_2}(k) - \mathbf{Adv}_A^{H_1}| < \mathbf{Adv}^{\text{ind-ecca}}_{\mathsf{TB\text{-}PKE},A}(k)$. For the latter, once again, decryption queries are handled as discussed above. The same bound is true for $\Pr_2[forge] - \Pr_1[forge]$.

Hybrid H_3: H_3 is the same as H_2 except that the key for the MAC in the challenge ciphertext (i.e. for computing c_3^*) is generated uniformly at random as opposed being set to $g(x^*)$.

We note that $|\mathbf{Adv}_A^{H_3}(k) - \mathbf{Adv}_A^{H_2}|$ is statistically bounded since the only information available about x^* is $H(x^*)$. But since H is compressing and g is pairwise-independent hash function, it operates as an extractor of the remaining randomness in x^* and outputs a uniformly random key for the MAC. Distinguishing a uniformly random key from $g(x^*)$ is therefore negligible and bounded by this statistical bound (see [8] for a complete argument). Decryption queries are handled as before. The same argument implies that $\Pr_3[forge] - \Pr_2[forge]$ is also negligible.

It remains for us show that $\Pr_3[forge]$ is also negligible but this automatically follows from the unforgeability of the MAC. ∎

5 Application to Adaptive Trapdoor Functions

We now give an application of ECCA-security to the theory of adaptive TDFs. Namely, we use it as a unifying concept to show that the notions of adaptive TDFs and tag-based adaptive ATDFs introduced by Kiltz *et al.* [29] are *equivalent* (via fully black-box reductions), resolving a foundational open question raised in [29]. To do so, we show below that both primitives are implied by *uniquely* randomness-recovering ECCA-secure PKE. Combined with Section 4, this shows that in fact uniquely randomness-recovering PKE, adaptive TDFs, and tag-based ATDFs are all equivalent.

5.1 From ECCA Security to Adaptivity

To construct an adaptive trapdoor function from a uniquely randomness-recovering ECCA-secure PKE scheme, we use part of the input to the former as coins for the latter used to encrypt the other part. Namely, let $\mathsf{PKE} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Rec})$ be a uniquely randomness-recovering public-key encryption scheme with message space $MsgSp$ and coin space $Coins$. Define a trapdoor function family $\mathsf{TDF}[\mathsf{PKE}] = (\mathsf{Tdg}, \mathsf{Eval}, \mathsf{Inv})$ on domain $MsgSp \times Coins$ as follows:

Alg $\mathsf{Tdg}(1^k)$	**Alg** $\mathsf{Eval}(pk, x)$	**Alg** $\mathsf{Inv}(sk, c)$
$(pk, sk) \leftarrow\!\!\$\ \mathsf{Kg}(1^k)$	$(m, r) \leftarrow x$	$m \leftarrow \mathsf{Dec}(sk, c)$
Return (pk, sk)	$c \leftarrow \mathsf{Enc}(pk, m; r)$	$r \leftarrow \mathsf{Rec}(sk, c)$
	Return c	Return (m, r)

Proposition 5.1 *Suppose* PKE *is uniquely randomness recovering and ECCA-secure. Then* $\mathsf{TDF}[\mathsf{PKE}]$ *defined above is adaptive one-way.*

Proof: Given an AOW-adversary I against $\mathsf{TDF}[\mathsf{PKE}]$, we can easily construct an ECCA-adversary $A = (A_1, A_2)$ against PKE, as follows:

Adversary $A_1^{\mathsf{Dec}^*(sk,\cdot)}(pk)$	**Adversary** $A_2^{\mathsf{Dec}^*(sk,\cdot)}(pk, c, St)$
$m_0, m_1 \leftarrow\!\!\$\ MsgSp(1^k)$	$m_0 \| m_1 \leftarrow St$
Return $(m_0, m_1, m_0 \| m_1)$	Run I on inputs pk, c:
	When I makes query y do:
	$(m, r) \leftarrow \mathsf{Dec}^*(sk, y)$
	Return (m, r)
	Let (m^*, r^*) be the output of I
	If $m_0 = m^*$ then return 0
	Else return 1

It is clear by construction that $\mathbf{Adv}^{\text{aow}}_{\mathsf{TDF},I}(\cdot) \leq \mathbf{Adv}^{\text{ind-ecca}}_{\mathsf{PKE},A}(\cdot)$ which proves the claim. ∎

5.2 From ECCA Security to Tag-Based Adaptivity

To construct a tag-based adaptive trapdoor function from a uniquely randomness-recovering ECCA-secure PKE scheme, we can use an analogous construction of [31, Section 4.4]. Namely, let $\mathsf{PKE} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Rec})$ be a uniquely randomness-recovering public-key encryption scheme. Define a tag-based trapdoor function family $\mathsf{TB\text{-}TDF}[\mathsf{PKE}] = (\mathsf{Tdg}, \mathsf{Eval}, \mathsf{Inv})$ as follows:

Alg $\mathsf{Tdg}(1^k)$	**Alg** $\mathsf{Eval}(pk, t, x)$	**Alg** $\mathsf{Inv}(sk, t, c)$
$(pk, sk) \leftarrow\!\!\$\ \mathsf{Kg}(1^k)$	$m \| r \leftarrow x$	$t' \| m \leftarrow \mathsf{Dec}(sk, c)$
Return (pk, sk)	$c \leftarrow \mathsf{Enc}(pk, t \| m; r)$	$r \leftarrow \mathsf{Rec}(sk, c)$
	Return c	If $t = t'$ then return $m \| r$
		Else return $\perp$

Proposition 5.2 *Suppose* PKE *is uniquely randomness-recovering and ECCA-secure. Then* $\mathsf{TB\text{-}TDF}[\mathsf{PKE}]$ *is tag-based adaptive one-way.*

Proof: Given an TB-AOW-adversary I against $\mathsf{TB\text{-}TDF}[\mathsf{PKE}]$, we construct an ECCA-adversary $A = (A_1, A_2, A_3)$ against PKE, as follows:

Adversary $A_1(pk)$	**Adversary** $A_1^{\mathsf{Dec}^*(sk,\cdot)}(pk, t)$	**Adversary** $A_2^{\mathsf{Dec}^*(sk,\cdot)}(pk, t, c, St)$
$t \leftarrow\!\!\$\ I_1(pk)$	$m_0, m_1 \leftarrow\!\!\$\ MsgSp(1^k)$	$m_0 \| m_1 \leftarrow St$
	Return $(m_0, m_1, m_0 \| m_1)$	Run I on inputs pk, t, c:
		When I makes query y, t' do:
		$(m, r) \leftarrow \mathsf{Dec}^*(sk, y, t')$
		Return (m, r)
		Let (m^*, r^*) be the output of I
		If $m_0 = m^*$ then return 0
		Else return 1

We claim that $\mathbf{Adv}^{\text{tb-aow}}_{\mathsf{TB\text{-}TDF},I}(\cdot) \leq \mathbf{Adv}^{\text{ind-ecca}}_{\mathsf{PKE},A}(\cdot)$. To see this, note that I_1, I_2 does make a query of the form $t' = t$, which by consistency of PKE means that A does not query its challenge ciphertext. ∎

6 Application to PKE with Non-Interactive Opening

In this section, we show that ECCA-secure encryption is a natural building-block for *public key encryption with non-interactive opening* (PKENO) [16, 15, 19, 20]. PKENO allows the receiver to non-interactively prove that a given ciphertext decrypts to a claimed message. Our constructions yield new and practical PKENO schemes. As discussed in the introduction, PKENO has applications to multiparty computation (*e.g.*, auctions and elections), secure message transmission, group signatures, and more.

PKENO. Public-key encryption with non-interactive opening (PKENO) extends a scheme $\mathsf{PKE} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ for public-key encryption by the following algorithms: Prove takes as input a secret key sk and a ciphertext c, and outputs a proof π. Ver takes as input a public key pk, a ciphertext c, a plaintext m and a proof π, and outputs 0 or 1. We write $\mathsf{PKENO} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Prove}, \mathsf{Ver})$.

We require *proof correctness*: for all ciphertexts (i.e. strings) c, the following is negligible:

$$\Pr[\mathsf{Ver}(pk, c, \mathsf{Dec}(sk, c), \mathsf{Prove}(sk, c)) \neq 1 \ : \ (pk, sk) \leftarrow\!\!\$\ \mathsf{Kg}(1^k)] \ .$$

Following [15, 19] we define security of PKENO by two notions, *indistinguishability under chosen-ciphertext and -proof attacks* (IND-CCPA) and *proof soundness*. The former guarantees that a ciphertext hides the plaintext even when the adversary can see decryptions of and proofs for other ciphertexts; the latter formalizes that no adversary should be able to produce a proof for a message and ciphertext that is not the encryption of that message. The formal definitions follow.

Experiment $\mathbf{Exp}^{\text{ind-ccpa}}_{\mathsf{PKENO},A}(k)$
$b \leftarrow\!\!\$\ \{0,1\}$; $(pk, sk) \leftarrow\!\!\$\ \mathsf{Kg}(1^k)$
$(m_0, m_1, St) \leftarrow\!\!\$\ A_1^{\mathsf{Dec}(sk,\cdot),\mathsf{Prove}(sk,\cdot)}(pk)$
$c \leftarrow\!\!\$\ \mathsf{Enc}(pk, m_b)$
$d \leftarrow\!\!\$\ A_2^{\mathsf{Dec}(sk,\cdot),\mathsf{Prove}(sk,\cdot)}(pk, c, St)$
If $d = b$ then return 1 else return 0

Experiment $\mathbf{Exp}^{\text{proof-snd}}_{\mathsf{PKENO},A}(k)$
$(pk, sk) \leftarrow\!\!\$\ \mathsf{Kg}(1^k)$
$(m', \pi', c') \leftarrow\!\!\$\ A(pk, sk)$
$m \leftarrow \mathsf{Dec}(sk, c')$
If $\mathsf{Ver}(pk, c', m', \pi') = 1$ and $m \neq m'$
then return 1 ; else return 0

Figure 2: Security experiments for PKENO.

Indistinguishability. Let $\mathsf{PKENO} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Prove}, \mathsf{Ver})$ be a public-key encryption scheme with non-interactive opening. We associate to PKENO, and an adversary $A = (A_1, A_2)$ the *chosen-ciphertext and -proof attack* experiment given on the left in Figure 2. We require that the output of A_1 satisfies $|m_0| = |m_1|$ and that A_2 does not query c to any of its oracles. We say that PKENO is *chosen-ciphertext and -proof-attack secure* (CCPA-secure) if $\mathbf{Adv}^{\text{ind-ccpa}}_{\mathsf{PKENO},A}(k) := 2 \cdot \Pr\left[\mathbf{Exp}^{\text{ind-ccpa}}_{\mathsf{PKENO},A}(k) \text{ outputs } 1\right] - 1$ is negligible for every efficient A.

Proof soundness. We associate to a scheme $\mathsf{PKENO} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Prove}, \mathsf{Ver})$ and an adversary $A = (A_1, A_2)$ a *proof-soundness* experiment, given on the right in Figure 2. We say that PKENO is *proof-sound* if $\mathbf{Adv}^{\text{proof-snd}}_{\mathsf{PKENO},A}(k) := \Pr\left[\mathbf{Exp}^{\text{proof-snd}}_{\mathsf{PKENO},A}(k) \text{ outputs } 1\right]$ is negligible for every efficient A.

We note that as compared to [15, 19] our definition of proof soundness also considers adversarially-produced ciphertexts, which need not even be a valid output of the encryption algorithm. Note that it is already required by proof correctness that the PKENO correctly proves decryption of such ciphertexts (which in general may or may not decrypt to $\perp$), so it would seem that constructions should achieve this stronger notion of proof soundness anyway.

Strong proof soundness. An even stronger notion of proof soundness is defined in [20], which also handles maliciously chosen public keys (*i.e.*, security for senders against a malicious receiver). Such a notion is quite challenging to achieve, and hence we mostly focus on the above formulation of proof soundness in the paper. However, in Appendix D we define notions of strong proof soundness and discuss how our constructions can be adapted to meet them.

6.1 PKENO-Compatible ECCA-Secure PKE

A natural approach to building PKENO suggested by [16] is to use a randomness-recovering encryption scheme and have the receiver provide the recovered coins as the proof. A moment's reflection reveals that for this approach to work, the encryption scheme must be ECCA secure in order to protect against chosen-proof attacks. In addition, as discussed in [16, 15, 20], we also need a way for the receiver to prove correct decryption of ciphertexts that are not in the range of the encryption algorithm, in which case such coins may not be defined. In this section we define a notion of *PKENO-compatible ECCA-secure encryption* for which we can do this. Below we define the properties such a scheme must have.

Partial-randomness recovery. It turns out that for such schemes we do not always achieve, nor need, the notion of full randomness recovery, so we first define a natural generalization we call *partial-randomness recovery*, which (loosely) says that enough of the random coins are recovered to uniquely identify the underlying message. (Such a notion is alluded to in [36], who note that their CCA-secure encryption is not actually fully randomness-recovering because they use a one-time signature, and is generally useful whenever we use a "publicly verifiable" but randomized component such as a one-time signature or NIZK.) However, in order to deal with the case that some ciphertexts outside the range of the encryption algorithm may not decrypt to $\perp$, we also *strengthen* what we get from randomness-recovering encryption in some respect; see the discussion following the definition.

Formally, Suppose Enc draws its coins from *Coins*. We say that a public-key encryption scheme $\mathsf{PKE} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ has *partial-randomness recovery* if it also has a *partial-randomness recovering algorithm*

pRec and a *message-consistency checking algorithm* Cons, which with the help of partial randomness can check whether a ciphertext c encrypts a message m. Namely:

- (Completeness) For all $(pk, sk) \leftarrow\!\!\$\ \mathsf{Kg}$ and all $c \in \{0,1\}^*, m \in \mathit{MsgSp}(1^k) \cup \{\perp\}$, let $s \leftarrow\!\!\$\ \mathsf{pRec}(sk, c)$:
$$\mathsf{Dec}(sk, c) = m \ \wedge \ m \neq \perp \ \Rightarrow \ \mathsf{Cons}(pk, c, m, s) = 1 \ .$$
- (Soundness) For all $(pk, sk) \leftarrow\!\!\$\ \mathsf{Kg}$ and all $c \in \{0,1\}^*, m \in \mathit{MsgSp}(1^k), s \in \{0,1\}^*$:
$$\mathsf{Cons}(pk, c, m, s) = 1 \ \Rightarrow \ \mathsf{Dec}(sk, c) = m \ .$$

In this case we say PKE is *ECCA-secure* if it is ECCA-secure as defined in Section 3 but where the Rec algorithm there is replaced with pRec.

It turns out that a fully randomness recovering scheme is not necessarily partial-randomness recovering as we have defined it. This is because the completeness condition requires that even for *invalid* ciphertexts c (*i.e.*, those that are never output by the encryption algorithm) that do not decrypt to $\perp$ but rather some $m \neq \perp$, enough partial randomness can still be recovered from c to check that it decrypts to m.

CIPHERTEXT VERIFIABILITY. We next define a notion of *ciphertext verifiability*, which intuitively means a verifier can check (with the help of some partial random coins) whether the decryption algorithm returns $\perp$ on a given ciphertext. Let $\mathsf{PKE} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{pRec}, \mathsf{Cons})$ be a public-key encryption scheme with partial-randomness recovery. We say that PKE has *ciphertext-verifiability* if it also has a *ciphertext-invalidity checking algorithm* Inval such that

- (Completeness) For all $(pk, sk) \leftarrow\!\!\$\ \mathsf{Kg}$ and all $c \in \{0,1\}^*$, let $s \leftarrow\!\!\$\ \mathsf{pRec}(sk, c)$. Then
$$\mathsf{Dec}(sk, c) = \perp \ \Rightarrow \ \mathsf{Inval}(pk, c, s) = 1 \ .$$
- (Soundness) For all $(pk, sk) \leftarrow\!\!\$\ \mathsf{Kg}$ and all $c \in \{0,1\}^*, s \in \{0,1\}^*$:
$$\mathsf{Inval}(pk, c, s) = 1 \ \Rightarrow \ \mathsf{Dec}(sk, c) = \perp \ .$$

We note that the notion of *public* ciphertext verifiability has been discussed informally in the literature and formalized and studied concurrently to our work by [34]. In a publicly verifiable scheme, the Inval algorithm would ignore the input s. Thus, our notion is more general. In this respect, it is noteworthy that our most efficient constructions using the symmetric-key version of the BCHK transform [8] are *not* publicly verifiable (although our basic constructions are).

PKENO-COMPATIBLE ECCA-SECURE PKE. We can now state our definition of PKENO compatibility. Let $\mathsf{PKE} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{pRec}, \mathsf{Cons}, \mathsf{Inval})$. We say that PKE is an *PKENO-compatible ECCA-secure PKE scheme* (or just PKENO-compatible for short) if it satisfies ECCA-security, partial-randomness recovery, and ciphertext verifiability as defined above. We next prove the main theorem of this section, namely that a PKENO-compatible ECCA-secure PKE scheme indeed gives us PKENO by using the idea of [16] described above.

PKENO CONSTRUCTION. Consider a PKENO-compatible ECCA encryption scheme $\mathsf{PKE} = (\mathsf{Kg}_{\mathrm{pke}}, \mathsf{Enc}_{\mathrm{pke}}, \mathsf{Dec}_{\mathrm{pke}}, \mathsf{pRec}_{\mathrm{pke}}, \mathsf{Cons}_{\mathrm{pke}}, \mathsf{Inval}_{\mathrm{pke}})$ and define $\mathsf{PKENO}[\mathsf{PKE}] = (\mathsf{Kg}_{\mathrm{pke}}, \mathsf{Enc}_{\mathrm{pke}}, \mathsf{Dec}_{\mathrm{pke}}, \mathsf{Prove}, \mathsf{Ver})$ with:

- $\mathsf{Prove}(sk, c)$: Output $s \leftarrow\!\!\$\ \mathsf{pRec}_{\mathrm{pke}}(sk, c)$.
- $\mathsf{Ver}(pk, c, m, \pi = s)$: If $m = \perp$ then return 1 iff $\mathsf{Inval}_{\mathrm{pke}}(pk, c, s) = 1$. Else return 1 iff $\mathsf{Inval}_{\mathrm{pke}}(pk, c, s) = 0$ and $\mathsf{Cons}_{\mathrm{pke}}(pk, c, m, s) = 1$.

Theorem 6.1 *Suppose* PKE *is a PKENO-compatible ECCA-secure encryption scheme. Then the construction* $\mathsf{PKENO}[\mathsf{PKE}]$ *above is a PKE scheme with non-interactive opening which is CCPA-secure and has proof soundness.*

Proof: For the case of proof correctness, we show that for $(pk, sk) \leftarrow\!\!\$\ \mathsf{Kg}(1^k)$ and for all strings $c \leftarrow \{0,1\}^*$:
$$\mathsf{Ver}(pk, c, \mathsf{Dec}(sk, c), \mathsf{Prove}(sk, c)) = 1 \ .$$

There are two cases: $\mathsf{Dec}(sk, c) = \perp$ and $\mathsf{Dec}(sk, c) \neq \perp$. If $\mathsf{Dec}(sk, c) = \perp$ then by completeness of ciphertext verifiability we have $\mathsf{Inval}(pk, c, \mathsf{pRec}(sk, c)) = 1$ hence $\mathsf{Ver}(pk, c, \mathsf{Dec}(sk, c), \mathsf{Prove}(sk, c)) = 1$. On the other hand, if $m = \mathsf{Dec}(sk, c) \neq \perp$ then by soundness of ciphertext verifiability (stating $\mathsf{Inval}(pk, c, s) = 1 \Rightarrow \mathsf{Dec}(sk, c) = \perp$) we have $\mathsf{Inval}(pk, c, s) = 0$. Moreover, by completeness of partial-randomness recovery we have $\mathsf{Cons}(pk, c, m, s) = 1$ and hence together: $\mathsf{Ver}(pk, c, \mathsf{Dec}(sk, c), \mathsf{Prove}(sk, c)) = 1$.

For proof soundness, in the relevant experiment in Figure 2, let (m', π', c') be the adversary's output and let $m \leftarrow \mathsf{Dec}(sk, c')$. Then it suffices to show that

$$m \neq m' \Rightarrow \mathsf{Ver}(pk, c', m', \pi') = 0 \ .$$

Again we distinguish two cases: (1) If $m' = \perp$ then by the definition of Ver we need to show that $\mathsf{Inval}(pk, c', \pi') = 0$. This is the case since $\perp = m' \neq m = \mathsf{Dec}(pk, c')$ means $\mathsf{Dec}(pk, c') \neq \perp$, which by soundness of ciphertext verifiability implies $\mathsf{Inval}(pk, c', s) = 0$ for all s, thus in particular for $s = \pi'$.
(2) If $m' \neq \perp$ then by definition of Ver it suffices to show that $\mathsf{Cons}(pk, c', m', \pi') = 0$. Since $m' \neq m = \mathsf{Dec}(sk, c')$, by soundness of partial-randomness recovery, we have $\mathsf{Cons}(pk, c', m', s) = 0$ for every s and thus in particular for $s = \pi'$.

IND-CCPA follows immediately from ECCA w.r.t. $\mathsf{pRec}_{\mathrm{pke}}$. The simulator can use its $\mathsf{pRec}_{\mathrm{pke}}$ oracle to simulate the Prove oracle. To simulate the Dec oracle on a query c, the simulator queries $\mathsf{pRec}_{\mathrm{pke}}$ for c to get s and outputs $\perp$ if $\mathsf{Inval}_{\mathrm{pke}}(pk, c, s) = 1$ and otherwise forwards a $\mathsf{Dec}_{\mathrm{pke}}$ oracle response. ∎

Tag-based. We also define a tag-based variant, namely *PKENO-compatible tag-based ECCA-secure encryption* $\mathsf{TB\text{-}PKE} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{pRec}, \mathsf{Cons}, \mathsf{Inval})$ analogously, where all algorithms except Kg now take an additional input t called the *tag*. The completeness and soundness definitions quantify over all tags t, and the scheme is required to satisfy the tag-based ECCA definition where Rec is replaced with pRec. One can convert any PKENO-compatible tag-based ECCA encryption scheme into a PEKNO-compatible (non-tag-based) ECCA encryption scheme by using (either version of) the BCHK transform [8], which we prove preserves both partial-randomness recovery and ciphertext verifiability. See Appendix C.

6.2 PKENO-Compatible PKE using Non-Interactive Zero-Knowledge

We show how to obtain PKENO-compatibility generically from any ECCA-secure randomness-recovering PKE by adding a non-interactive zero-knowledge proof (NIZK) of ciphertext "well-formedness." The approach of using a NIZK originates from [15, 20], although not with respect to ECCA-secure encryption. We note that we do not require the starting ECCA-secure encryption scheme to be *uniquely* randomness-recovering (although our constructions in Section 4 achieve this), but it should have perfect correctness.

Construction. Consider a RR ECCA-secure PKE scheme $\mathsf{PKE} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Rec}, \mathsf{Cons})$ with perfect correctness and a simulation-sound NIZK proof system $\mathsf{NIZK} = (\mathsf{Setup}, \mathsf{Prv}, \mathsf{Vrf})$ for the language $\mathcal{L} := \{(pk, c) \,|\, \exists (m, r) \,:\, c = \mathsf{Enc}(pk, m; r)\}$. We define a partial-randomness-recovering scheme $\mathsf{PKE}_{\mathrm{prr}} = (\mathsf{Kg}_{\mathrm{prr}}, \mathsf{Enc}_{\mathrm{prr}}, \mathsf{Dec}_{\mathrm{prr}}, \mathsf{pRec}_{\mathrm{prr}}, \mathsf{Cons}_{\mathrm{prr}}, \mathsf{Inval}_{\mathrm{prr}})$ as follows:

- $\mathsf{Kg}_{\mathrm{prr}}(1^k)$: Run $crs \leftarrow\!\!\$\ \mathsf{Setup}(1^k)$; $(pk, sk) \leftarrow\!\!\$\ \mathsf{Kg}(1^k)$. Output $\overline{pk} := (crs, pk)$ and $\overline{sk} := (crs, pk, sk)$.
- $\mathsf{Enc}_{\mathrm{prr}}((crs, pk), m; r)$: Set $c := \mathsf{Enc}(pk, m; r)$ and $\tau \leftarrow\!\!\$\ \mathsf{Prv}(crs, (pk, c), (m, r))$; output $\overline{c} := (c, \tau)$.
- $\mathsf{Dec}_{\mathrm{prr}}((crs, pk, sk), (c, \tau))$: If $\mathsf{Vrf}(crs, (pk, c), \tau) = 0$ then output $\perp$; else output $\mathsf{Dec}(sk, c)$.
- $\mathsf{pRec}_{\mathrm{prr}}((crs, pk, sk), (c, \tau))$: If $\mathsf{Vrf}(crs, (pk, c), \tau) = 0$ then output $\perp$; else output $\mathsf{Rec}(sk, c)$.
- $\mathsf{Cons}_{\mathrm{prr}}((crs, pk), (c, \tau), m, s)$: Return 1 iff $\mathsf{Vrf}(crs, (pk, c), \tau) = 1$ and $c = \mathsf{Enc}(pk, m; s)$.
- $\mathsf{Inval}_{\mathrm{prr}}((crs, pk), (c, \tau), s)$: Return 1 iff $\mathsf{Vrf}(crs, (pk, c), \tau) = 0$.

Proposition 6.2 *Suppose* PKE *is randomness-recovering with perfect correctness and ECCA-secure and* NIZK *is simulation-sound and zero-knowledge. Then* $\mathsf{PKE}_{\mathrm{prr}}[\mathsf{PKE}, \mathsf{NIZK}]$*, defined above, is an ECCA-secure PKE scheme with partial-randomness recovery and public verifiability (i.e., it is PKENO-compatible).*

Note that the NIZK here is required to be simulation-sound [39]. Intuitively, the reason is that upon receiving challenge ciphertext (c^*, τ^*), the adversary might (legitimately) submit some (c^*, τ') to its decryption oracle for $\tau' \neq \tau^*$.

Proof: We first show that the scheme $\mathsf{PKE}_{\text{prr}}$ is still ECCA-secure (with respect to the partial randomness recovery function pRec). Let A be an adversary against ECCA security of $\mathsf{PKE}_{\text{prr}}$. We use A to break ECCA security of the underlying scheme PKE. Upon receiving our challenge c^*, we simulate a NIZK proof τ^* to create a challenge (c^*, τ^*) for A. Consider A making an ECCA-query Dec^* for (c, τ): if τ is invalid on c, we return $\perp$; else we forward c to our own oracle and give A the reply. Note that simulation soundness of the NIZK implies that A cannot produce a query (c^*, τ) for a $\tau \neq \tau^*$ which is valid for c^*. (This is formally proven analogously to the proof of full anonymity of the group signature construction in [4], where a group signature is defined as a CCA-secure ciphertext and a simulation-sound NIZK proof of well-formedness of the ciphertext.) This means that every query that A makes can be forwarded to our own oracle.

It remains to show completeness and soundness of both partial randomness recovery and ciphertext verifiability, as defined in Section 6.1. For all these notions, let $(\overline{pk} = (crs, pk), \overline{sk} = (crs, pk, sk))$ be the output of Kg_{prr}.

Completeness of randomness recovery: Let $\overline{c} \in \{0,1\}^*$ and let $s \leftarrow\!\!\$\ \mathsf{pRec}_{\text{prr}}(\overline{sk}, \overline{c})$. Suppose $\mathsf{Dec}_{\text{prr}}(\overline{sk}, \overline{c}) = m$ with $m \neq \perp$. By the definition of $\mathsf{Dec}_{\text{prr}}$, with $(c, \tau) \leftarrow \overline{c}$, we have $\mathsf{Vrf}(crs, (pk, c), \tau) = 1$ $(*)$. This means that $\mathsf{pRec}_{\text{prr}}(\overline{sk}, (c, \tau))$ outputs $s \leftarrow \mathsf{Rec}(sk, c)$, which by perfect correctness of PKE recovers coins s such that $c = \mathsf{Enc}(pk, m; s)$. Together with $(*)$ this implies $\mathsf{Cons}_{\text{prr}}(\overline{pk}, (c, \tau), m, s) = 1$.

Soundness of partial-randomness recovery: Let $(c, \tau) \in \{0,1\}^*, m \in \mathit{MsgSp}(1^k)$ and $s \in \{0,1\}^*$. Suppose $\mathsf{Cons}_{\text{prr}}(\overline{pk}, (c, \tau), m, s) = 1$, that is $\mathsf{Vrf}(crs, (pk, c), \tau) = 1$ $(*)$ and $c = \mathsf{Enc}(pk, m; s)$. By perfect correctness of PKE this implies $\mathsf{Dec}(sk, c) = m$, which together with $(*)$ yields: $\mathsf{Dec}_{\text{prr}}(\overline{sk}, (c, \tau)) = m$.

Completeness of ciphertext verifiability: Let $(c, \tau) \in \{0,1\}^*$ and let $s \leftarrow\!\!\$\ \mathsf{pRec}_{\text{prr}}(\overline{sk}, c)$. Suppose that $\mathsf{Dec}_{\text{prr}}(\overline{sk}, (c, \tau)) = \perp$. Then we have either (1) $\mathsf{Vrf}(crs, (pk, c), \tau) = 0$ or (2) $\mathsf{Dec}(sk, c) = \perp$. We show (2) implies (1) by contraposition: If Vrf outputs 1 then by soundness of NIZK there exist $(m, r) : c = \mathsf{Enc}(pk, m; r)$. By perfect correctness of PKE we have $\mathsf{Dec}(sk, c) = m \neq \perp$. In either case we therefore have $\mathsf{Vrf}(crs, (pk, c), \tau) = 0$ and thus $\mathsf{Inval}_{\text{prr}}(\overline{pk}, (c, \tau), s) = 1$.

Soundness of ciphertext verifiability: Let $(c, \tau) \in \{0,1\}^*$ and $s \in \{0,1\}^*$. Assume $\mathsf{Inval}_{\text{prr}}(\overline{pk}, (c, \tau), s) = 1$, that is, $\mathsf{Vrf}(crs, (pk, c), \tau) = 0$. Then by definition of $\mathsf{Dec}_{\text{prr}}$, we have $\mathsf{Dec}_{\text{prr}}(\overline{sk}, (c, \tau)) = \perp$. ∎

7 Efficient PKENO Constructions

7.1 Construction from DLIN Using Groth-Sahai

We first give a general construction of PKENO-compatible tag-based PKE from tag-based ATDFs that come equipped with a NIZK proof system for perimage existence. Note that this is different from (weaker than) a proof of ciphertext well-formedness as used in Section 6.2.

PKENO-Compatible Tag-based PKE Construction. The construction essentially follows Section 4.3, except for incorporation of the NIZK. Let $\mathsf{TB\text{-}TDF} = (\mathsf{Tdg}, \mathsf{Eval}, \mathsf{Inv})$ be a tag-based trapdoor function family with tag space TagSp and hardcore bit hc. Let $\mathsf{NIZK} = (\mathsf{Setup}_{\text{nizk}}, \mathsf{Prv}_{\text{nizk}}, \mathsf{Vrf}_{\text{nizk}})$ be a NIZK proof system for the language $\mathcal{L} := \{(ek, t, y) \,|\, \exists x : y = \mathsf{Eval}(ek, t, x)\}$. We define tag-based PKE scheme $\mathsf{TB\text{-}PKE} = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{pRec}, \mathsf{Cons}, \mathsf{Inval})$ with tag space TagSp and message space $\{0,1\}^\ell$ as follows. Algorithm Kg outputs $\overline{pk} = (crs, ek), \overline{dk} = (crs, td)$ where $crs \leftarrow\!\!\$\ \mathsf{Setup}_{\text{nizk}}(1^k)$ and $(ek, td) \leftarrow\!\!\$\ \mathsf{Tdg}(1^k)$, and the remaining algorithms are defined via:

Alg $\mathsf{Enc}((crs, ek), t, m)$
  For $i = 1$ to ℓ do:
    $x_i \leftarrow\!\!\$\ \{0,1\}^k$; $c_{i,1} \leftarrow \mathsf{Eval}(ek, t, x_i)$
    $c_{i,2} \leftarrow \mathsf{hc}(x) \oplus m[i]$
    $\tau_i \leftarrow\!\!\$\ \mathsf{Prv}_{\text{nizk}}(crs, (ek, t, c_{i,1}), x_i)$
  $\mathbf{c} \leftarrow ((c_{1,1}, c_{1,2}, \tau_1), \ldots, (c_{\ell,1}, c_{\ell,2}, \tau_\ell))$
  Return $\mathbf{c}$

Alg $\mathsf{pRec}((crs, td), t, \mathbf{c})$
  $((c_{1,1}, c_{1,2}, \tau_1), \ldots, (c_{\ell,1}, c_{\ell,2}, \tau_\ell)) \leftarrow \mathbf{c}$
  For $i = 1$ to ℓ do:
    If $\mathsf{Vrf}_{\text{nizk}}(crs, (ek, t, c_{i,1}), \tau_i) = 0$ then return $\perp$
    $x_i \leftarrow \mathsf{Inv}(td, t, c_{i,1})$
  Return $(x_1, \ldots, x_\ell)$

Alg $\mathsf{Inval}((crs, ek), t, \mathbf{c}, \mathbf{s})$
  $((c_{1,1}, c_{1,2}, \tau_1), \ldots, (c_{\ell,1}, c_{\ell,2}, \tau_\ell)) \leftarrow \mathbf{c}$
  For $i = 1$ to ℓ do:
    If $\mathsf{Vrf}_{\text{nizk}}(crs, (ek, t, c_{i,1}), \tau_i) = 0$ then return 1
  Return 0

Alg $\mathsf{Dec}((crs, td), t, \mathbf{c})$
  $((c_{1,1}, c_{1,2}, \tau_1), \ldots, (c_{\ell,1}, c_{\ell,2}, \tau_\ell)) \leftarrow \mathbf{c}$
  For $i = 1$ to ℓ do:
    If $\mathsf{Vrf}_{\text{nizk}}(crs, (ek, t, c_{i,1}), \tau_i) = 0$
      then return $\perp$
    $x_i \leftarrow \mathsf{Inv}(td, t, c_{i,1})$
    $m[i] \leftarrow \mathsf{hc}(x_i) \oplus c_{i,2}$
  Return m

Alg $\mathsf{Cons}((crs, ek), t, \mathbf{c}, m, \mathbf{s})$
  $((c_{1,1}, c_{1,2}, \tau_1), \ldots, (c_{\ell,1}, c_{\ell,2}, \tau_\ell)) \leftarrow \mathbf{c}$
  $\mathbf{s} \leftarrow (x_1, \ldots, x_\ell)$
  For $i = 1$ to ℓ do:
    If $\mathsf{Vrf}_{\text{nizk}}(crs, (ek, t, c_{i,1}), \tau_i) = 0$
      then return 0
    If $\mathsf{Eval}(ek, t, x_i) \neq c_{i,1}$ then return 0
    If $\mathsf{hc}(x_i) \oplus m[i] \neq c_{i,2}$ then return 0
  Return 1

Proposition 7.1 *Suppose* TB-TDF *is a tag-based adaptive TDF and* NIZK *is zero-knowledge. Then* TB-PKE *as defined above is a PKENO-compatible tag-based PKE scheme.*

Interestingly, we do not require the NIZK to be simulation-sound here, in contrast to the generic construction in Section 6.2. Intuitively, this because both primitives (*i.e.*, TB-ATDF and PKENO-compatible tag-based PKE) are tag-based, so the adversary cannot submit (parts of) the challenge ciphertext to its Dec^* oracle, since the tag must be different from t chosen at the beginning of the game.

Proof: We first need to show that TB-PKE is ECCA-secure (with respect to the partial randomness recovery function pRec). The proof is analogous to that of Proposition 4.11 (which in turn is similar to that of Claim 4.1) and works by a hybrid argument and a reduction to adaptive one-wayness of TB-TDF. The difference is that we have added the NIZK proofs, which are however not needed for ECCA security. Therefore, in a first game hop, we replace the proofs τ_i contained in the challenge ciphertext with simulated proofs. By zero-knowledge of NIZK, this game is indistinguishable from the original game $\mathbf{Exp}^{\text{ind-tb-ecca}}_{\mathsf{TB\text{-}PKE}}$. Now that the proofs are constructed without using the witnesses x_i, we can simulate the game in the reduction to adaptive one-wayness of TB-TDF, using our Inv oracle to answer the adversary's Dec^* queries.

It remains to show completeness and soundness of both partial randomness recovery and ciphertext verifiability, as defined in Section 6.1. In the following, let $((crs, ek), (crs, td))$ be the output of $\mathsf{Kg}(1^\lambda)$.

Completeness of randomness recovery: Let $\mathbf{c} \in \{0,1\}^*$ and let $\mathbf{s} \leftarrow\!\!\$\ \mathsf{pRec}((crs, td), t, \mathbf{c})$. Suppose $\mathsf{Dec}((crs, td), t, \mathbf{c}) = m$ with $m \neq \perp$; thus with $((c_{1,1}, c_{1,2}, \tau_1), \ldots, (c_{\ell,1}, c_{\ell,2}, \tau_\ell)) \leftarrow \mathbf{c}$, for all $1 \leq i \leq \ell$: (i) $\mathsf{Vrf}_{\text{nizk}}(crs, (ek, t, c_{i,1}), \tau_i) = 1$ and (ii) $m[i] = \mathsf{hc}(\mathsf{Inv}(td, t, c_{i,1})) \oplus c_{i,2}$. By (i) we have that $\mathsf{pRec}(crs, td), t, \mathbf{c})$ returns $x_1, \ldots, x_\ell$ with $\mathsf{Eval}(ek, t, x_i) = c_{i,1}$, which together with (i) and (ii) means that $\mathsf{Cons}((crs, ek), t, \mathbf{c}, m, \mathbf{s}) = 1$.

Soundness of partial-randomness recovery: Let $((c_{1,1}, c_{1,2}, \tau_1), \ldots, (c_{\ell,1}, c_{\ell,2}, \tau_\ell)) \in \{0,1\}^*, m \in \{0,1\}^\ell$, $(x_1, \ldots, x_\ell) \in \{0,1\}^*$ and suppose $\mathsf{Cons}((crs, ek), t, \mathbf{c}, m, \mathbf{s}) = 1$. That is, for all $1 \leq i \leq \ell$: $(*)$ $\mathsf{Vrf}_{\text{nizk}}(crs, (ek, c_{i,1}), \tau_i) = 1$, $\mathsf{Eval}(ek, t, x_i) = c_{i,1}$ and $\mathsf{hc}(x_i) \oplus m[i] = c_{i,2}$. Thus $x_i = \mathsf{Inv}(td, t, c_{i,1})$ and $\mathsf{hc}(x_i) \oplus c_{i,2} = m[i]$ and thus together with $(*)$: $\mathsf{Dec}((crs, ek), t, \mathbf{c}) = m$.

Completeness of ciphertext verifiability: Let $\mathbf{c} \in \{0,1\}^*$ and $\mathbf{s} \leftarrow\!\!\$\ \mathsf{pRec}((crs, td), t, \mathbf{c})$; suppose that $\mathsf{Dec}((crs, ek), t, \mathbf{c}) = \perp$. Then with $((c_{1,1}, c_{1,2}, \tau_1), \ldots, (c_{\ell,1}, c_{\ell,2}, \tau_\ell)) \leftarrow \mathbf{c}$, for some $1 \leq i \leq \ell$: $\mathsf{Vrf}_{\text{nizk}}(crs, (ek, t, c_{i,1}), \tau_i) = 0$; thus $\mathsf{Inval}((crs, ek), t, \mathbf{c}, \mathbf{s}) = 1$.

Soundness of ciphertext verifiability: Let $\mathbf{c} \in \{0,1\}^*$ and $\mathbf{s} \in \{0,1\}^*$; assume $\mathsf{Inval}((crs, ek), t, \mathbf{c}, \mathbf{s}) = 1$. That is, with $((c_{1,1}, c_{1,2}, \tau_1), \ldots, (c_{\ell,1}, c_{\ell,2}, \tau_\ell)) \leftarrow \mathbf{c}$, for some $1 \le i \le \ell$: $\mathsf{Vrf}_{\mathrm{nizk}}(crs, (ek, t, c_{i,1}), \tau_i) = 0$. Then by definition of Dec, we have $\mathsf{Dec}((crs, td), t, \mathbf{c}) = \perp$. ∎

TB-ATDF FROM LOSSY+ABO TDF. To realize the tag-based ATDF in the above construciton, we will specfically use the tag-based ATDF from lossy+ABO TDF given by [29]. We we briefly recall the construction here (and refer the reader to [29] for formal details and the definitions of the relevant primitives). Suppose we have a lossy TDF LF and an all-but one (ABO) TDF $\mathsf{ABO\text{-}F}$. The construction also uses a hash function $T\colon \{0,1\}^n \to \{0,1\}^k$ for some $n = n(k)$, which is required to be (target) collision resistant. An evaluation key for the constructed tag-based ATDF is a pair of keys $(ek_{\mathrm{ltf}}, ek_{\mathrm{abo}})$ for LF and $\mathsf{ABO\text{-}F}$; the trapdoor is the corresponding pair $(td_{\mathrm{ltf}}, td_{\mathrm{abo}})$. The evaluation of ATDF on input $x \in \{0,1\}^k$ and tag $t \in \{0,1\}^n$ under key $(ek_{\mathrm{ltf}}, ek_{\mathrm{abo}})$ is defined as (y_1, y_2) with

$$y_1 \leftarrow \mathsf{LF}(ek_{\mathrm{ltf}}, x) \qquad y_2 \leftarrow \mathsf{ABO\text{-}F}(T(t), ek_{\mathrm{abo}}, x)$$

Finally, inversion, given (y_1, y_2), tag t, and trapdoor $(td_{\mathrm{ltf}}, td_{\mathrm{abo}})$, computes $x_1 \leftarrow \mathsf{LF}^{-1}(td_{\mathrm{ltf}}, y_1)$ and $x_2 \leftarrow \mathsf{ABO\text{-}F}^{-1}(td_{\mathrm{abo}}, t, y_2)$. If $x_1 = x_2$ it returns the common value, otherwise $\perp$.

A "GS-FRIENDLY" INSTANTIATION. Groth-Sahai (GS) proofs [25] are efficient NIZKs without random oracles for a certain class of statements over bilinear groups. Here we provide an instantiation of the above TB-ATDF construction for which preimage existence is a GS statement; we specifically use the lossy and ABO TDFs of Freeman *et al.* [18] based on the decision linear (DLIN) assumption. We sketch the construction here, omitting the details not relevant to show that GS proofs can be used to show that there exists a preimage.

We first describe the ABO-TDF. In a group $\mathbb{G}$ of order p, generated by $g \in \mathbb{G}$, the scheme is defined as follows. To sample a function with a lossy branch b^*, choose a matrix $A \leftarrow\!\!\$\ \mathbb{Z}_p^{n\times n}$ with rank 1 and define $M := A - b^* I_n$, where I_n is the identity matrix. Define $\mathbf{S} \in \mathbb{G}^{n\times n}$ as the matrix with components $S_{ij} := g^{m_{ij}}$, where m_{ij} are the components of M. The function $f_{\mathbf{S},b}\colon \{0,1\}^n \to \mathbb{G}^n$ indexed by $\mathbf{S}$ is evaluated on a branch b is defined as:

$$f_{\mathbf{S},b}(\vec{x}) := (\textstyle\prod_{j=1}^n S_{ij}^{x_j} \cdot g^{b\cdot x_i})_{i=1}^n\,.$$

The same holds for the LTDF, which is a special case of the above for $b = 0$. Let us denote the corresponding TB-ATDF obtained by plugging in these instantiations as $\mathsf{TB\text{-}TDF}_{dlin}$.

Proposition 7.2 *For $\mathsf{TB\text{-}TDF}_{dlin}$, the corresponding language $\mathcal{L} := \{(ek, t, y) \mid \exists x : y = \mathsf{Eval}(ek, t, x)\}$ can be expressed as a pairing-product equation over bilinear group elements, which is a statement in the language of Groth-Sahai proofs.*

Proof: For the above, it suffices to show how to prove that there exists a preimage for a value y under $\mathsf{ABO\text{-}F}$ for a certain branch b and a key ek. Given a function value $(F_i)_{i=1}^n \in \mathbb{G}^n$, we need to show that there exists $(x_i)_{i=1}^n \in \{0,1\}^n$ s.t. $F_i = \prod S_{ij}^{x_j} \cdot g^{b\cdot x_i}$ for every $1 \le i \le n$. Instead of giving a direct proof, we show that the projections $x_i \mapsto X_i = g^{x_i}$ of x_i to $\mathbb{G}$ satisfy the following equations (where e denotes the bilinear map) for every $1 \le i \le n$:

$$e(X_i, X_i \cdot g^{-1}) = e(g, g^0) \qquad\qquad e(g, F_i) = \textstyle\prod_{j=1}^n e(S_{ij}, X_j) \cdot e(g^b, X_i)$$

As the above are pairing-product equations over the variables $X_1, \ldots X_n$, we can use Groth-Sahai proofs to show satisfiability, that is, to show that there exist $X_1, \ldots, X_n \in \mathbb{G}$, which satisfy all the equations simultaneously: Let $(X_i)_{i=1}^n$ satisfy the above equations and let $(\xi_i)_{i=1}^n \in \mathbb{Z}_p^n$ be such that $X_i = g^{\xi_i}$, for all $1 \le i \le n$; then the first equation ensures that $\xi_i \in \{0,1\}$ (since we have $e(g,g)^{\xi_i\cdot(\xi_i-1)} = e(g,g)^0$). The second equation ensures that $F_i = \prod S_{ij}^{\xi_j} \cdot g^{b\cdot\xi_i}$. Together this yields that $(\xi_i)_{i=1}^n$ is a preimage of $(F_i)_{i=1}^n$. ∎

7.2 Construction from Instance-Independent RSA

We show the RSA-based TB-TDF of [29] provides range verifiability automatically, avoiding the extra cost of executing NIZK.

II-RSA BASED TB-ATDF. We briefly recall the construction and refer to [29] for more details. An evaluation key is a random RSA modulus $N = pq$ and the trapdoor is (p, q). The construction also uses a collision-resistant hash function $H\colon \{0,1\}^n \to \mathcal{P}_\ell$ for some $n = n(k), \ell = \ell(k)$, where $\mathcal{P}_\ell$ is the set of ℓ-bit prime numbers. (In theory, such a hash function can actually be built without computational assumption [32].) The tag space is $\{0,1\}^n$ and input space is $\mathbb{Z}_N^*$. The evaluation on N, tag t, and input x is $x^{H(t)} \bmod N$. Inversion on inputs $(p, q), t, y$, computes $y^s \bmod N$ where $s \leftarrow H(t)^{-1} \bmod \phi(N)$ (which can be computed efficiently given p, q). Security relies on the *instance-independent RSA assumption (II-RSA)*; see [29] for the definition.

TAG-BASED PKE SCHEME. We plug the above TB-ATDF into the construction of Section 7.1, with the modification that there will be no invocation of $\mathsf{Prv}_{\mathrm{nizk}}$ (the proof is an empty string) and $\mathsf{Vrf}_{\mathrm{nizk}}$ is replaced a the simple check of whether $c < N$ or not. Denote this resulting tag-based PKE scheme by $\mathsf{TB\text{-}PKE}_{iirsa}$.

Proposition 7.3 $\mathsf{TB\text{-}PKE}_{iirsa}$ *described above is a PKENO-compatible tag-based PKE scheme under the II-RSA assumption.*

Proof: (Sketch.) The security of the scheme then directly follows from proof of proposition 7.1 as anyone can check preimage existence of a point c relative to evaluation key N by checking whether $c < N$. Note that we need to relax our PKENO definitions to achieve completeness and soundness only for PPT generated inputs and fail with negligible probability, since a PPT adversary may produce a c in $\mathbb{Z}_N \setminus \mathbb{Z}_N^*$ with negligible probability assuming factoring N is hard. (For simplicity, we do not make these relaxations in our formal definitions.) ∎

EFFICIENCY. The PKENO scheme that results from this instantiation (using proposition 6.1) is quite efficient, requiring just one modular exponentiation to encrypt. Note that for efficiency, one can relax the need to hash to primes by using "division-intractible" hashing instead, as defined in [21]. This could be instantiated by using a cryptographic hash function with roughly 1024-bit output [13]. In terms of efficiency our scheme is comparable to a previous DLIN-based PKENO construction of [20], which requires roughly 5 elliptic curve exponentiations to encrypt (but is secure under a more standard assumption). We leave a more detailed efficiency comparison among these schemes to future work.

Acknowledgements

We are grateful to Mihir Bellare, whose comments and suggestions improved our results and presentation. We also thank Eike Kiltz for his involvement in the early stages of this work. Work done in part while the first author was at Microsoft Research, New England and the fourth author was at Boston University.

References

[1] M. Bellare. Private communication, 2012. 6

[2] 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, Aug. 1998. 4

[3] M. Bellare, D. Hofheinz, and S. Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In A. Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 1–35. Springer, Apr. 2009. 4

[4] M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In E. Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 614–629. Springer, May 2003. 3, 22

[5] M. Bellare and A. Palacio. Towards plaintext-aware public-key encryption without random oracles. In P. J. Lee, editor, *ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 48–62. Springer, Dec. 2004. 4

[6] M. Bellare and P. Rogaway. Optimal asymmetric encryption. In A. D. Santis, editor, *EUROCRYPT'94*, volume 950 of *LNCS*, pages 92–111. Springer, May 1994. 4

[7] M. Bellare and S. Yilek. Encryption schemes secure under selective opening attack. Cryptology ePrint Archive, Report 2009/101, 2009. `http://eprint.iacr.org/`. 4

[8] D. Boneh, R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. *SIAM Journal on Computing*, 36(5):1301–1328, 2007. 3, 16, 17, 20, 21, 30

[9] R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively secure multi-party computation. In *28th ACM STOC*, pages 639–648. ACM Press, May 1996. 4

[10] R. Canetti, H. Krawczyk, and J. B. Nielsen. Relaxing chosen-ciphertext security. In D. Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 565–582. Springer, Aug. 2003. 4

[11] R. Canetti, H. Lin, and R. Pass. Adaptive hardness and composable security in the plain model from standard assumptions. In *51st FOCS*, pages 541–550. IEEE Computer Society Press, Oct. 2010. 4

[12] S. G. Choi, D. Dachman-Soled, T. Malkin, and H. Wee. Black-box construction of a non-malleable encryption scheme from any semantically secure one. In R. Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 427–444. Springer, Mar. 2008. 9, 11

[13] J.-S. Coron and D. Naccache. Security analysis of the Gennaro-Halevi-Rabin signature scheme. In B. Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 91–101. Springer, May 2000. 25

[14] 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. 28, 29

[15] I. Damgård, D. Hofheinz, E. Kiltz, and R. Thorbek. Public-key encryption with non-interactive opening. In T. Malkin, editor, *CT-RSA 2008*, volume 4964 of *LNCS*, pages 239–255. Springer, Apr. 2008. 1, 2, 18, 19, 21

[16] I. Damgård and R. Thorbek. Non-interactive proofs for integer multiplication. In M. Naor, editor, *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 412–429. Springer, May 2007. 2, 3, 18, 19, 20

[17] C. Dwork, M. Naor, O. Reingold, and L. J. Stockmeyer. Magic functions. *Journal of the ACM*, 50(6):852–921, 2003. 4

[18] D. M. Freeman, O. Goldreich, E. Kiltz, A. Rosen, and G. Segev. More constructions of lossy and correlation-secure trapdoor functions. In P. Q. Nguyen and D. Pointcheval, editors, *PKC 2010*, volume 6056 of *LNCS*, pages 279–295. Springer, May 2010. 3, 24

[19] D. Galindo. Breaking and repairing Damgård et al. public key encryption scheme with non-interactive opening. In M. Fischlin, editor, *CT-RSA 2009*, volume 5473 of *LNCS*, pages 389–398. Springer, Apr. 2009. 2, 18, 19

[20] D. Galindo, B. Libert, M. Fischlin, G. Fuchsbauer, A. Lehmann, M. Manulis, and D. Schröder. Public-key encryption with non-interactive opening: New constructions and stronger definitions. In D. J. Bernstein and T. Lange, editors, *AFRICACRYPT 10*, volume 6055 of *LNCS*, pages 333–350. Springer, May 2010. 2, 3, 18, 19, 21, 25, 31

[21] R. Gennaro, S. Halevi, and T. Rabin. Secure hash-and-sign signatures without the random oracle. In J. Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 123–139. Springer, May 1999. 25

[22] R. Gennaro and V. Shoup. A note on an encryption scheme of Kurosawa and Desmedt. Cryptology ePrint Archive, Report 2004/194, 2004. `http://eprint.iacr.org/`. 16

[23] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *21st ACM STOC*, pages 25–32. ACM Press, May 1989. 10

[24] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984. 5

[25] J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In N. P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Apr. 2008. 3, 24

[26] D. Hofheinz and E. Kiltz. Secure hybrid encryption from weakened key encapsulation. In A. Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 553–571. Springer, Aug. 2007. 4, 28, 29

[27] S. Hohenberger, A. B. Lewko, and B. Waters. Detecting dangerous queries: A new approach for chosen ciphertext security. In *EUROCRYPT 2012*, LNCS, pages 663–681. Springer, 2012. 2, 4, 9, 10, 11, 12

[28] S. A. Kakvi, E. Kiltz, and A. May. Certifying rsa. In *ASIACRYPT*, pages 404–414, 2012. 31

[29] E. Kiltz, P. Mohassel, and A. O'Neill. Adaptive trapdoor functions and chosen-ciphertext security. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 673–692. Springer, May 2010. 1, 2, 3, 8, 9, 15, 16, 17, 24, 25, 27, 28

[30] P. D. MacKenzie, M. K. Reiter, and K. Yang. Alternatives to non-malleability: Definitions, constructions, and applications (extended abstract). In *TCC*, pages 171–190, 2004. 2

[31] P. D. MacKenzie, M. K. Reiter, and K. Yang. Definitions, constructions, and applications (extended abstract). In M. Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 171–190. Springer, Feb. 2004. 5, 18

[32] S. Micali, M. O. Rabin, and S. P. Vadhan. Verifiable random functions. In *40th FOCS*, pages 120–130. IEEE Computer Society Press, Oct. 1999. 25

[33] S. Myers and A. Shelat. Bit encryption is complete. In *50th FOCS*, pages 607–616. IEEE Computer Society Press, Oct. 2009. 1, 2, 9

[34] J. M. G. Nieto, M. Manulis, B. Poettering, J. Rangasamy, and D. Stebila. Publicly verifiable ciphertexts. In *SCN*, pages 393–410, 2012. 20

[35] R. Pass, abhi shelat, and V. Vaikuntanathan. Construction of a non-malleable encryption scheme from any semantically secure one. In C. Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 271–289. Springer, Aug. 2006. 9

[36] C. Peikert and B. Waters. Lossy trapdoor functions and their applications. In R. E. Ladner and C. Dwork, editors, *40th ACM STOC*, pages 187–196. ACM Press, May 2008. 1, 3, 5, 8, 19

[37] C. Rackoff and D. R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In J. Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 433–444. Springer, Aug. 1992. 1

[38] A. Rosen and G. Segev. Chosen-ciphertext security via correlated products. In O. Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 419–436. Springer, Mar. 2009. 1, 8

[39] A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th FOCS*, pages 543–553. IEEE Computer Society Press, Oct. 1999. 22

[40] H. Wee. Efficient chosen-ciphertext security via extractable hash proofs. In T. Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 314–332. Springer, Aug. 2010. 1

[41] A. C. Yao. Theory and applications of trapdoor functions. In *23rd FOCS*, pages 80–91. IEEE Computer Society Press, Nov. 1982. 27

A Standard Primitives

Trapdoor functions. A *trapdoor function family* [41] is a triple of algorithms $\mathsf{TDF} = (\mathsf{Tdg}, \mathsf{Eval}, \mathsf{Inv})$. The key-generation algorithm Tdg returns an evaluation key ek and matching trapdoor td. The deterministic evaluation algorithm Eval takes ek and $x \in \{0,1\}^k$ to return an image y. The deterministic inversion algorithm Inv takes td and y to return a point x. We require that for all $x \in \{0,1\}^k$,

$$\Pr[\mathsf{Inv}(td, \mathsf{Eval}(ek, x)) = x \,:\, (ek, td) \leftarrow\!\!\$\, \mathsf{Tdg}(1^k)]$$

is 1. (Note that this implies the trapdoor function is *injective*.) We say that TDF is *tag-based* [29] with tag-space $TagSp$ if $\mathsf{Eval}, \mathsf{Inv}$ take an additional input $t \in TagSp$ called the *tag* and for all $x \in \{0,1\}^k$ and $t \in TagSp(1^k)$,

$$\Pr[\mathsf{Inv}(td, t, \mathsf{Eval}(ek, t, x)) = x \,:\, (ek, td) \leftarrow\!\!\$\, \mathsf{Tdg}(1^k)]$$

is probability 1.

AUTHENTICATED ENCRYPTION. We use the definition of [26] for authenticated encryption. An authenticated symmetric encryption (AE) scheme $\mathsf{AE} = (\mathsf{AE.Enc}, \mathsf{AE.Dec})$ is specified by its encryption algorithm $\mathsf{AE.Enc}$ (encrypting $m \in \mathit{MsgSp}(k)$ with a key $K \in K(k)$) and decryption algorithm $\mathsf{AE.Dec}$ (returning $m \in \mathit{MsgSp}(k)$ or $\perp$). Here we restrict ourselves to deterministic $\mathsf{AE.Enc}$ and $\mathsf{AE.Dec}$. The AE scheme needs to provide privacy (indistinguishability against one-time attacks) and authenticity (ciphertext authenticity against one-time attacks). This is simultaneously captured by defining the *ae-ot-advantage* of an adversary A as:

$$\mathbf{Adv}^{\text{ot-ae}}_{A,\mathsf{AE}}(k) = 2 \cdot \Pr[b = b' \,:\, K \leftarrow\!\!\$\, K(k)\,;\, b \leftarrow\!\!\$\, \{0,1\}\,;\, b' \leftarrow\!\!\$\, A^{\mathsf{LoR}_b(\cdot,\cdot),\mathsf{DoR}_b(\cdot)}(1^k)] - 1\ .$$

Here, $\mathsf{LoR}_b(m_0, m_1)$ returns $\mathsf{AE.Enc}(K, m_b)$, and A is allowed only one query to this left-or-right encryption oracle, with a pair of equal length messages. Furthermore, the decrypt-or-reject oracle $\mathsf{DoR}_0(C)$ returns $m \leftarrow \mathsf{AE.Dec}(K, C)$ and $\mathsf{DoR}_1(C)$ always returns $\perp$ (reject). A is allowed only one query to this oracle which must be different from the output of the left-or-right oracle. An encryption scheme is a *one-time authenticated encryption* if $\mathbf{Adv}^{\text{ot-ae}}_{A,\mathsf{AE}}(k)$ is negligible for any PPT adversary A.

KEY ENCAPSULATION MECHANISMS. The KEM/DEM paradigm was first formalized in [14]. We borrow our formal definitions from [26]. A key-encapsulation mechanism $\mathsf{KEM} = (\mathsf{KEM.kg}, \mathsf{KEM.enc}, \mathsf{KEM.enc})$ with key-space $K(k)$ consists of three polynomial-time algorithms. Via $(pk, sk) \leftarrow\!\!\$\, \mathsf{KEM.kg}(1^k)$ the randomized key-generation algorithm produces public/secret keys for security parameter k; via $(K, C) \leftarrow\!\!\$\, \mathsf{KEM.enc}(pk)$, the randomized encapsulation algorithm creates a uniformly distributed symmetric key $K \in K(k)$ together with a ciphertext C; via $K \leftarrow \mathsf{KEM.dec}(sk, C)$ the possessor of secret key sk decrypts ciphertext C to get back a key K which is an element in K or a special rejection symbol $\perp$. For consistency, we require that for all all $(K, C) \leftarrow\!\!\$\, \mathsf{KEM.enc}(pk)$ we have $\Pr[\mathsf{KEM.dec}(sk, C) = K] = 1$, where the probability is taken over the choice of $(pk, sk) \leftarrow\!\!\$\, \mathsf{KEM.kg}(1^k)$, and the coins of all the algorithms in the expression above. Here we only consider KEMs that produce perfectly uniformly distributed keys (i.e., we require that for all public keys pk that can be output by $\mathsf{KEM.kg}$, the first component of $\mathsf{KEM.enc}(pk)$ has uniform distribution).

Experiment $\mathbf{Exp}^{\text{kem-cca}}_{\mathsf{KEM},A}(k)$
$b \leftarrow\!\!\$\, \{0,1\}\,;\, (pk, sk) \leftarrow\!\!\$\, \mathsf{KEM.kg}(1^k)$
$K^*_0 \leftarrow\!\!\$\, K(k);\, (K^*_1, C^*) \leftarrow\!\!\$\, \mathsf{KEM.enc}(pk)$
$d \leftarrow\!\!\$\, A_1^{\mathsf{KEM.dec}^*(sk,\cdot)}(pk, K^*_b, C^*)$
If $d = b$ then return 1 else return 0

Oracle $\mathsf{KEM.dec}^*(sk, C)$
$K \leftarrow \mathsf{KEM.dec}(sk, C)$
Return K

Above we require that A does not query c^* to its oracle. Define the *kem-cca advantage* of A against KEM as

$$\mathbf{Adv}^{\text{ekm-cca}}_{\mathsf{KEM},A}(k) = 2 \cdot \Pr\left[\, \mathbf{Exp}^{\text{kem-cca}}_{\mathsf{KEM},A}(k) \text{ outputs } 1 \,\right] - 1\ .$$

We say that KEM is *chosen-ciphertext secure* if $\mathbf{Adv}^{\text{kem-cca}}_{\mathsf{KEM},A}(\cdot)$ is negligible for every efficient A.

B From ATDF to ECCA via KEM/DEM

The above constructions assume that the ATDF/tb-ATDF only provides us with a single hardcore bit. But, if a linear number of hardcore bits are available (e.g. based on Lossy TDFs as discussed in [29]), one can design significantly more efficient ECCA PKE constructions. The basic idea is simple: we use the ATDF/tb-ATDF with linear hardcore bits to encrypt a one-time secret key k via a key encapsulation mechanism (KEM), and use k to encrypt the message via a data encapsulation mechanism (DEM). The standard KEM/DEM paradigm guarantees that the resulting hybrid PKE scheme is CCA secure if the

KEM component and the DEM component are both CCA secure[7] (e.g. see [14]).

In our case, however, we need the hybrid PKE to be ECCA secure and randomness recovering as well. One can construct a KEM based on a ATDF by simply using the hardcore bits as the one-time key. It is easy to see that this construction is both ECCA secure and randomness recovering. The natural next step is to use a CCA (or ECCA) DEM component to obtain a hybrid PKE with the desired properties.

CCA security of DEM is not sufficient. Surpringly, this does not work: hybrid encryption does *not* preserve the ECCA security of the KEM. Roughly speaking, the subtlety in the proof arises when the simulator needs to answer decryption queries for ciphertexts that have the same KEM component as the challenge ciphertext but a different DEM component. In the standard proof, such decryption queries are answered by decrypting the DEM component and returning the message (without having to decrypt the KEM component). But, to achieve ECCA security, we need to return all the randomness to adversary, including those used in the KEM component. To solve this we instead use a a one-time *authenticated encryption* scheme as the DEM, in which case we show the resulting hybrid PKE is ECCA secure and randomness recovering.

An ECCA KEM/DEM Construction from ATDFs. Consider the following construction based on any ATDF with linear hardcore bits. Let $\mathsf{TDF} = (\mathsf{Tdg}, \mathsf{Eval}, \mathsf{Inv})$ be a trapdoor function with a hardcore function hc, and $\mathsf{AE} = (\mathsf{AE.Kg}, \mathsf{AE.Enc}, \mathsf{AE.Dec})$ be a deterministic authenticated encryption scheme. Define the following multi-bit public-key encryption scheme $\mathsf{PKE}[\mathsf{TDF}] = (\mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$:

Alg $\mathsf{Kg}(1^k)$	**Alg** $\mathsf{Enc}(ek, m)$	**Alg** $\mathsf{Dec}(td, ((y_1, y_2), \mathsf{flag}))$
$(ek, td) \leftarrow\!\!\$\ \mathsf{Tdg}(1^k)$	$x \leftarrow\!\!\$\ \{0,1\}^k$	If $\mathsf{flag} = 1$ then return (y_1, y_2)
Return (ek, td)	$y_1 \leftarrow \mathsf{Eval}(ek, x)$	Else $K \leftarrow \mathsf{hc}(\mathsf{Inv}(td, y_1))$
	$y_2 \leftarrow \mathsf{AE.Enc}(\mathsf{hc}(x), m)$	Return $(x, \mathsf{AE.Dec}(k, y_2))$
	Return (y_1, y_2)	

Proposition B.1 *Suppose* TDF *is adaptive one-way and* AE *is a one-time authenticated encryption. Then* $\mathsf{PKE}[\mathsf{TDF}]$ *defined above is ECCA-secure and randomness-recovering.*

Proof: It is easy to see that the above construction is randomness-recovering. In particular, the decryption algorithm recovers $x = \mathsf{Inv}(td, y_1)$, computes $K = \mathsf{hc}(x)$ and then uses K to recover the message encrypted in y_2. Since the AE is deterministic, there is no additional randomness to recover.

Next, we show that $\mathsf{PKE}[\mathsf{TDF}]$ is also ECCA. We take advantage of the deferred analysis technique in this proof as well. Consider the following sequence of games:

Hybrid H_0: The first game is the *ind-cca* experiment for $\mathsf{PKE}[\mathsf{TDF}]$. Denote the challenge ciphertext by $c^* = (c_1^*, c_2^*)$, and the secret key used to encrypt c_2^* by k^*.

Hybrid H_1: H_1 is the same as H_0 except that on decryption queries of the form $c = (c_1^*, c_2)$ by the adversary we return $\perp$.

Note that $|\mathbf{Adv}_A^{H_0}(k) - \mathbf{Adv}_A^{H_1}(k)| < \Pr_1[valid]$, where *valid* is the event that in game H_1, for some decryption query $c \neq c^*$ where $c_1 = c_1^*$, c is a valid ciphertext. Assuming no decryption error, this probability is bounded by $\Pr_1[forge]$ where *forge* is the event that c_2 is valid ciphertext for the AE scheme. We postpone the analysis of the bound on $\Pr_1[forge]$ until the final hybrid.

Hybrid H_2: H_2 is the same as H_1, except that in the challenge ciphertext, we use a uniformly random key K' for the AE scheme.

Lets denote by KEM, the KEM component of the above construction. It is easy to see that $|\mathbf{Adv}_A^{H_2}(k) - \mathbf{Adv}_A^{H_1}(k)| \leq \mathbf{Adv}^{\text{kem-cca}}_{B,\mathsf{KEM}}$. Note that the same bound is true for $|\Pr_2[forge] - \Pr_1[forge]|$.

[7] It is possible to relax the security requirement for the KEM component but the CCA security of the DEM seems necessary in order to obtain a CCA secure PKE (see [26]).

Hybrid H_3: H_3 is the same as H_2, except that instead of encrypting m_b for the challenge ciphertext (in the DEM component), we encrypt $0^{|m_0|}$.

Obviously, $\mathbf{Adv}_A^{H_3}(k) = 1/2$. It is also not hard to see that $|\mathbf{Adv}_A^{H_3}(k) - \mathbf{Adv}_A^{H_2}(k)| \leq \mathbf{Adv}_{C,\mathsf{AE}}^{\text{ot-ae}}(k)$. The same bound is true for $|\Pr_3[forge] - \Pr_2[forge]|$ as well.

The last thing we need to show is that $\Pr_3[forge]$ is negligible, but now the key K' for the AE is generated at random, we have that $\Pr_3[forge] < \mathbf{Adv}_{C,\mathsf{AE}}^{\text{ot-ae}}(k)$, hence concluding the proof.

∎

An ECCA KEM/DEM Construction from tb-ATDFs. The above construction can also be used to yield an ECCA tag-based PKE from any tb-ATDF with essentially an identical proof. Then we can apply the transformation of Section 4.3 based on [8], to turn this into a standard ECCA PKE scheme (see proposition 4.13).

C PKENO-Compatible Tag-based PKE to PKENO-Compatible PKE

We show that starting with a PKENO-compatible ECCA tag-based PKE, the construction of Section 4.3 based on the BCHK transformation [8] yields a PKENO-compatible ECCA encryption scheme. We focus here on the "more-efficient" version of the BCHK transform that uses symmetric primitives; the simpler version with one-time signatures also works, however.

Let $\mathsf{TB\text{-}PKE} = (\mathsf{Kg}_{tag}, \mathsf{Enc}_{tag}, \mathsf{Dec}_{tag}, \mathsf{pRec}_{tag}, \mathsf{Inval}_{tag})$ be a partially randomness-recovering ciphertext-verifiable tag-based public-key encryption scheme, H, g be hash functions as in Section 4.3, and $\mathsf{MAC} = (\mathsf{mac}, \mathsf{ver})$ be a message-authentication code. We define a partial-randomness-recovering scheme $\mathsf{PKE}_{\text{pce}} = (\mathsf{Kg}_{\text{pce}}, \mathsf{Enc}_{\text{pce}}, \mathsf{Dec}_{\text{pce}}, \mathsf{Cons}_{\text{pce}}, \mathsf{Inval}_{\text{pce}})$ as follows:

- $\mathsf{Kg}_{\text{pce}}(1^k)$: $(pk, sk) \leftarrow\!\!\$\ \mathsf{Kg}_{tag}(1^k)$; output (pk, sk).
- $\mathsf{Enc}_{\text{pce}}(pk, m; r||x)$: $c_1 \leftarrow H(x)$; $c_2 \leftarrow\!\!\$\ \mathsf{Enc}_{tag}(pk, c_1, m\|x; r)$; $c_3 \leftarrow \mathsf{mac}(g(x), c_2)$; output (c_1, c_2, c_3).
- $\mathsf{Dec}_{\text{pce}}(sk, (c_1, c_2, c_3))$: $m\|x \leftarrow \mathsf{Dec}_{tag}(sk, c_1, c_2)$; if $H(x) \neq c_1$ or $\mathsf{ver}(g(x), c_2, c_3) = 0$ then output $\perp$; else output m.
- $\mathsf{pRec}_{\text{pce}}((pk, sk), (c_1, c_2, c_3))$: $m\|x \leftarrow\!\!\$\ \mathsf{Dec}_{tag}(sk, c_2)$; $r \leftarrow\!\!\$\ \mathsf{pRec}_{tag}(sk, c_1, c_2)$; output (x, r).
- $\mathsf{Cons}_{\text{pce}}(pk, (c_1, c_2, c_3), m, (x, r))$: Return 1 iff $H(x) = c_1$, $\mathsf{ver}(g(x), c_2, c_3) = 1$ and $\mathsf{Cons}_{tag}(pk, c_1, c_2, m\|x, r) = 1$.
- $\mathsf{Inval}_{\text{pce}}(pk, (c_1, c_2, c_3), (x, r))$: Return 1 iff $H(x) \neq c_1$ or $\mathsf{ver}(g(x), c_2, c_3)) \neq 1$, or $\mathsf{Inval}_{tag}(pk, c_1, c_2, r) = 1$.

Proposition C.1 *If* $\mathsf{TB\text{-}PKE}$ *is a PKENO-compatible ECCA tag-based PKE scheme then* $\mathsf{PKE}_{\text{pce}}$ *defined above is a PKENO-compatible ECCA encryption scheme.*

Proof: We have already proven the ECCA security of the scheme in Proposition 4.13. In order to prove that $\mathsf{PKE}_{\text{pce}}$ is a PKENO-compatible ECCA encryption scheme we must show that it satisfies partial-randomness recovery and ciphertext verifiability.

We begin with partial-randomness recovery (pRR). Recall that completeness for pRR is defined as follows: For all $(pk, sk) \leftarrow\!\!\$\ \mathsf{Kg}_{\text{pce}}$ and all $c = (c_1, c_2, c_3) \in \{0,1\}^*, m \in \mathit{MsgSp}(1^k) \cup \{\perp\}$, let $(x, r) \leftarrow\!\!\$\ \mathsf{pRec}_{\text{pce}}(sk, (c_1, c_2, c_3))$. Then

$$\mathsf{Dec}_{\text{pce}}(sk, c) = m \wedge m \neq \perp \ \Rightarrow \mathsf{Cons}_{\text{pce}}(pk, c, m, s) = 1 \ .$$

Let $m\|x \leftarrow\!\!\$\ \mathsf{Dec}_{tag}(sk, c_2)$. Then by definition of $\mathsf{Dec}_{\text{pce}}$ that when $\mathsf{Dec}_{\text{pce}}$ returns $m \neq \perp$ we have that $H(x) = c_1$ and $\mathsf{ver}(g(x), c_2, c_3) = 1$. Additionally, by pRR completeness of $\mathsf{TB\text{-}PKE}$, we have that r is returned by $\mathsf{pRec}_{tag}(sk, c_1, c_2)$, and thus by pRR completeness of $\mathsf{TB\text{-}PKE}$ that $\mathsf{Cons}_{tag}(pk, c_1, c_2, m\|x, r) = 1$. Together we have that $\mathsf{Cons}_{\text{pce}}(pk, (c_1, c_2, c_3), m, (x, r))$ outputs 1.

Next, recall that soundness for partial randomness recovery (pRR) is defined as follows: For all $(pk, sk) \leftarrow\!\!\$\ \mathsf{Kg}_{\mathrm{pce}}$ and all $c = (c_1, c_2, c_3) \in \{0,1\}^*, m \in \mathit{MsgSp}(1^k), s = (x, r) \in \{0,1\}^*$:

$$\mathsf{Cons}_{\mathrm{pce}}(pk, c, m, s) = 1 \ \Rightarrow\ \mathsf{Dec}_{\mathrm{pce}}(sk, c) = m \ .$$

Soundness for partial randomness recovery follows immediately from the same notion for TB-PKE by the definitions of $\mathsf{Cons}_{\mathrm{pce}}$ and $\mathsf{Dec}_{\mathrm{pce}}$.

We now move onto showing the ciphertext verifiability property. Recall that completeness for ciphertext verifiability is defined as follows: For all $(pk, sk) \leftarrow\!\!\$\ \mathsf{Kg}_{\mathrm{pce}}$ and all $c = (c_1, c_2, c_3) \in \{0,1\}^*$, let $(x, r) \leftarrow\!\!\$\ \mathsf{pRec}_{\mathrm{pce}}(sk, c)$. Then

$$\mathsf{Dec}_{\mathrm{pce}}(sk, c) = \perp \ \Rightarrow\ \mathsf{Inval}_{\mathrm{pce}}(pk, c, (x, r)) = 1 \ .$$

If $\mathsf{Dec}_{\mathrm{pce}}(sk, c) = \perp$ then we must have that $H(x) \neq c_1$ or $\mathsf{Ver}_{tag}(pk, c_1, c_2) \neq 1$, or $\mathsf{Dec}_{tag}(sk, c_1, c_2) = \perp$. Completeness for ciphertext verifiability of TB-PKE implies that in the last case $\mathsf{Inval}_{tag}(pk, c_1, c_2, r) = 1$. Together this implies that $\mathsf{Inval}_{\mathrm{pce}}$ returns 1.

Recall that soundness for ciphertext verifiability is defined as follows: For all $(pk, sk) \leftarrow\!\!\$\ \mathsf{Kg}_{\mathrm{pce}}$ and all $c = (c_1, c_2, c_3) \in \{0,1\}^*, s = (x, r) \in \{0,1\}^*$:

$$\mathsf{Inval}_{\mathrm{pce}}(pk, (c_1, c_2, c_3), (x, r)) = 1 \ \Rightarrow\ \mathsf{Dec}_{\mathrm{pce}}(sk, (c_1, c_2, c_3)) = \perp \ .$$

Note that if $\mathsf{Inval}_{\mathrm{pce}}(pk, c, s) = 1$ we have that $H(x) \neq c_1$ or $\mathsf{ver}(g(x), c_2, c_3)) \neq 1$ or $\mathsf{Inval}_{tag}(pk, c_1, c_2, r) = 1$. In the last case, by soundness of TB-PKE, we have $\mathsf{Dec}_{tag}(sk, c_1, c_2) = \perp$. Together this implies that $\mathsf{Dec}_{\mathrm{pce}}(sk, (c_1, c_2, c_3)) = \perp$. ∎

D Achieving Strong Proof Soundness

The following two notions, given in [20], strengthen proof soundness. Consider the following two games:

Experiment $\mathbf{Exp}^{\text{s-proof-snd}}_{\mathsf{PKENO},A}(k)$
$(pk, m, St) \leftarrow\!\!\$\ A_1(1^k)$
$c \leftarrow\!\!\$\ \mathsf{Enc}(pk, m)$
$(m', \pi') \leftarrow\!\!\$\ A_2(pk, c, St)$
If $m \in \mathit{MsgSp}$, $\mathsf{Ver}(pk, c, m', \pi') = 1$ and $m \neq m'$
then return 1 ; else return 0

Experiment $\mathbf{Exp}^{\text{s-comm}}_{\mathsf{PKENO},A}(k)$
$(pk, c, m, \pi, m', \pi') \leftarrow\!\!\$\ A(1^k)$
If $\mathsf{Ver}(pk, c, m, \pi) = 1$ and
$\mathsf{Ver}(pk, c, m', \pi') = 1$ and
$m \neq m'$ then return 1
Else return 0

We say that PKENO is *strongly proof-sound* if for every efficient $A = (A_1, A_2)$ the probability of $\mathbf{Exp}^{\text{s-proof-snd}}_{\mathsf{PKENO},A}$ outputting 1 is negligible. Moreover, PKENO is *strongly committing* if for every efficient A the probability of $\mathbf{Exp}^{\text{s-comm}}_{\mathsf{PKENO},A}$ outputting 1 is negligible.

These notions of proof soundness strengthen the original ones in that they let the adversary choose the public key. Note that we cannot base such security on standard schemes, as for an adversarially chosen public-key there might not even exist a decryption key.

Efficient scheme based on II-RSA. However, for our efficient PKENO scheme based on II-RSA in Section 7.2, this problem can be overcome by having the verifier check that the RSA function defined by a ciphertext is a permutation. That is, in the PKENO scheme, for strong soundness algorithm $\mathsf{Ver}(N, c', m', \pi')$ can be modified to first check that $(N, H(t))$ define a permutation, where t is the corresponding tag obtained from the ciphertext c' (depending on which version of the BCHK transform is used, t will be e.g. a verification key for a one-time signature). In particular, Kakvi *et al.* [28] shows how to efficiently verify that RSA parameters (N, e) indeed define a permutation as long as $e \geq N^{1/4+\varepsilon}$. Thus, this check can be done efficiently if we hash to at least, say, 600 bits (for a 2048-bit modulus).