Public Key Encryption Resilient to Post-challenge Leakage and Tampering Attacks

Suvradip Chakraborty(✉) and C. Pandu Rangan

Department of Computer Science and Engineering, Indian Institute of Technology Madras, Chennai, India
{suvradip,rangan}@cse.iitm.ac.in

Abstract. In this paper, we introduce a new framework for constructing public-key encryption (PKE) schemes resilient to joint *post-challenge/after-the-fact* leakage and tampering attacks in the bounded leakage and tampering (BLT) model, introduced by Damgård et al. (Asiacrypt 2013). All the prior formulations of PKE schemes considered leakage and tampering attacks only *before* the challenge ciphertext is made available to the adversary. However, this restriction seems necessary, since achieving security against post-challenge leakage and tampering attacks in its full generality is impossible, as shown in previous works. In this paper, we study the post-challenge/after-the-fact security for PKE schemes against bounded leakage and tampering under a restricted yet meaningful and reasonable notion of security, namely, the *split-state leakage and tampering model*. We show that it is possible to construct secure PKE schemes in this model, tolerating arbitrary (but bounded) leakage and tampering queries; thus overcoming the previous impossibility results.

To this end, we formulate a new notion of security, which we call *entropic post-challenge* IND-CCA-BLT secure PKE. We first define a weaker notion called *entropic restricted post-challenge* IND-CCA-BLT secure PKE, which can be instantiated using the (standard) DDH assumption. We then show a generic compiler from our entropic restricted notion to the entropic notion of security using a simulation-extractable non-interactive zero-knowledge argument system. This requires an untamperable common reference string, as in previous works. Finally, we demonstrate the usefulness of our entropic notion of security by giving a simple and generic construction of post-challenge IND-CCA-BLT secure PKE scheme in the split-state leakage and tampering model. This also settles the *open problem* posed by Faonio and Venturi (Asiacrypt 2016).

Keywords: After-the-fact · Post-challenge · Entropic PKE · Split-state · Memory tampering · Related-key attacks · Bounded leakage and tampering

M. Matsui (Ed.): CT-RSA 2019, LNCS 11405, pp. 23–43, 2019.
https://doi.org/10.1007/978-3-030-12612-4_2

1 Introduction and Related Works

Traditionally, cryptographic schemes have been analyzed assuming that an adversary only have *black-box access* to the underlying functionality, and in no way is allowed to manipulate the internal state of the functionality. Leakage and tamper-resilient cryptography studies on designing secure protocols and primitives against an adversary who goes way beyond black-box access to protocol algorithms and gets information by directly accessing/tampering the memory or the internal computations of the system. These physical attacks can be broadly categorized into *passive* and *active* attacks. In case of passive attacks, the adversary tries to recover information via some *side-channel attacks* that include timing measurements, power analysis, electromagnetic measurements, microwave attacks, memory attacks and many more [15,17,18]. In case of active attacks, the adversary can modify the secret data/key of a targeted cryptographic scheme by applying various physical attacks, and later violate the security of the primitive by observing the effect of such changes at the output. These classes of attacks are called *memory tampering attacks* or *related key attacks* (RKA). These attacks can be launched both in software or hardware, like, injecting faults in the device, altering the internal power supply or clock of the device, or shooting the chip with a laser etc.

The formal study of security of cryptosystems, in particular block ciphers, against related key attacks was initiated by Bellare and Kohno [3]. In their setting, the adversary can continuously tamper with the secret key of the cryptosystem by choosing tampering functions from a restricted class of functions. One might hope to provably resist a cryptosystem against arbitrary efficiently computable tampering functions. Unfortunately, this type of *unrestricted tampering* is shown to be impossible by Gennaro et al. [13], without making further assumptions, like self-destruct mechanism, where the device simply blows up and erases all its intermediate values (including the secret key) after an tampering attempt is detected by the device. One useful line of research is to investigate the security of cryptosystems against *restricted* classes of tampering attacks. In most of these schemes, it is assumed that the secret key belongs to some finite field, and the allowed modifications consists of linear or affine functions, or all polynomial of bounded degree applied to the secret key.

Another interesting line of research was initiated in Asiacrypt 2013 by Damgård et al. [8], which is called the model of *bounded tampering.* In this model, the adversary is allowed to make a *bounded* number of tampering queries, however, there is no further restriction on the functions, unlike the previous works. Note that this model of bounded unrestricted tampering is orthogonal to the model of continuous but restricted tampering model of [3]. In [8], the authors showed a construction of signature scheme (in the random oracle model) and public-key encryption scheme (in the standard model) in the bounded leakage and tampering (BLT) model, where, apart from bounded unrestricted tampering, the adversary is also allowed to obtain bounded leakage from the secret key of the cryptosystem. Faonio and Venturi [12] later improved the state-of-the-art for the construction of signature schemes (in the standard model) and

PKE scheme (without involving pairings and zero-knowledge proofs) in the BLT model.

In all the above constructions of PKE schemes [8,12], the adversary is allowed to make *only* pre-challenge tampering queries. In other words, the adversary can specify a bounded number (say τ) of tampering queries T_i ($i \in [\tau]$) before the challenge phase, and gets access to the tampered decryption oracle $\mathsf{Dec}(\widetilde{sk}_i, \cdot)$, where $\widetilde{sk}_i = T_i(sk)$. However, after receiving the challenge ciphertext, the adversary is not allowed to make even a single tampering query. This severely restricts the meaning and applicability of the existing security notions and that of the resulting constructions of the cryptographic primitives satisfying these notions. In particular, this means that even if the adversary tampers with the secret key/memory only once, the secrecy of all the previously encrypted messages before that tampering attempt cannot be guaranteed. However, note that, this is not a limitation of the existing security notions or the constructions. Indeed, as shown in [16,20], tolerating *post-challenge* (also called *after-the-fact*) tampering in it full generality is *impossible*. In particular, the adversary could simply overwrite the secret key depending on the bit b that is encrypted in the challenge ciphertext c^*, and thus gain some advantage in guessing the value of b by asking additional decryption queries. We refer the reader to [8, Sect. 4.4] for the detailed attack. The above impossibility result holds even if the adversary is allowed to make even a single post-challenge tampering query followed by a single decryption query (with respect to the original secret key). Similar impossibility result is known to hold for the setting of leakage as well, in the sense that even if the adversary obtains a single bit of leakage in the post-challenge phase, this is enough to completely break the security of the PKE scheme. This is because the adversary can simply encode the decryption function with the challenge ciphertext and the two challenge messages in the leakage function and obtain exactly the bit b that the challenger tries to hide.

Halevi and Lin [16] addressed this issue of after-the-fact leakage, and defined an appropriate security model, namely the *split-state* leakage model (more on this below), and showed how to construct semantically-secure PKE scheme under this restricted security model. This was later extended to handle CCA security under the same split-state leakage model in [5,23]. However, note that, for the case of tampering, there are no suitable security notions or definitions to handle post-challenge tampering. This definitional problem was acknowledged in the prior works [8,12]. However, no solution to this issue was offered. Indeed it is mentioned in [12] that "it remains open how to obtain CCA security for PKE against "*after-the-fact*" tampering and leakage, where both tampering and leakage can still occur after the challenge ciphertext is generated".

1.1 Our Contributions and Techniques

In this work, we study post-challenge/after-the-fact leakage and tampering attacks in the context of public-key encryption. As discussed above, achieving resilience to post challenge tampering attack in its most general form is impossible. To this end, we formulate an appropriate security model that avoids the impossibility result shown in [8], and at the same time enables secure and efficient construction of PKE schemes in our new model. Our approach to the solution is *modular* in nature and is also surprisingly *simple*. In particular, we show how to effectively (and in a non-trivial way) combine together the appropriate works from the domain of leakage and tamper-resilience to arrive at our current solution. We discuss more on this below.

Split-State Leakage and Tampering Model: We draw the motivation of our work from that of Halevi and Lin [16]. To take care of after-the-fact leakage, the authors in [16] considered the *split-state leakage* model, where the secret key of the cryptosystem is split into multiple disjoint parts, and the adversary can observe (arbitrary) bounded leakage from each of these parts, but in an independent fashion. In order to take care of leakage and tampering jointly, we consider the *split-state leakage and tampering* model. Similar to the split-state leakage model, this model also considers the case where the secret key is also split into multiple disjoint parts (in our case only two, and hence optimal) and the adversary can obtain independent leakages from each of these parts. In addition, the adversary is also allowed to tamper each of the secret key components/parts independently. Note that, the split-state tampering model is already a very useful and widely used model and it captures bit tampering and block-wise tampering attacks, where the adversary can tamper each bit or each block of the secret key independently. The split-state tampering model is also well studied in the context of non-malleable codes [1,10,11], where similar type of impossibility results hold. We then proceed to construct our PKE scheme in this model. Lastly, one may note that, in the post-challenge setting in the context of a PKE scheme, the adversary may specify a tampering function to be an identity function and get the challenge ciphertext decrypted under the original secret keys (even in split-state model), and trivially win the security game. To avoid this, we enforce the condition that, when the adversary queries the (tampered) decryption oracle with the challenge ciphertext, the tampered keys need to be different from the original secret key. In other words, the post-challenge tampering functions must not be identity functions with respect to the challenge ciphertext[1].

Entropic Restricted Post-challenge IND-CCA-BLT PKE: We first formulate a new notion of *entropic restricted post-challenge* IND-CCA-BLT-*secure* PKE scheme. Our notion can be seen as an *entropic version* of the notion of restricted (pre-challenge) IND-CCA-BLT secure PKE of Damgård et al. [8], augmented with post challenge leakage and tampering queries. The definition of restricted IND-CCA-BLT-security [8] says that the adversary is given access to a *restricted* (faulty) decryption oracle, i.e., it is allowed to query only valid ciphertexts to the tampered decryption oracles (as opposed to any arbitrary ciphertexts as in the full fledged IND-CCA-BLT security game). Note that, in the definition

[1] However, note that, the tampering functions may be identity functions with respect to ciphertexts $c \neq c^*$, where c^* is the challenge ciphertext. This also emulates access to the (original) decryption oracle to the adversary.

of [8], the adversary is allowed to make only pre-challenge leakage and tampering queries. Our notion of entropic restricted post-challenge IND-CCA-BLT security captures the following intuition: Suppose we sample a message M from a high min-entropy distribution. Given a ciphertext encrypting M, and even given (bounded) leakage from the secret key and access to a restricted (tampered) decryption oracle (even if both leakage and tampering happens after observing the challenge ciphertext), the message M still retains enough min-entropy in it. We then show that the cryptosystem of Boneh et al. [4] (referred to as BHHO cryptosystem) satisfies our entropic restricted notion. The main idea of our construction is the leakage to tamper reduction for the BHHO cryptosystem as shown in [8]. Note that, using leakage to simulate tampering is non-trivial, since for each tampered secret key the adversary can make polynomially many (tampered) decryption oracle queries. Hence the amount of key-dependent information that the adversary receives cannot be simulated by a small amount of (bounded) leakage. However, as shown in [8], in case of BHHO cryptosystem for each (pre-challenge) tampering query it is possible to simulate polynomially many decryption queries under it by just leaking a single group element, thus reducing tampering to leakage. We use similar ideas and show that the BHHO cryptosystem with appropriate parameters satisfy our entropic restricted notion of security, even if leakage and tampering is allowed in the post-challenge phase. We note that, the work of Faonio and Venturi [12] gives a comparatively efficient construction of IND-CCA-BLT secure PKE scheme compared to the work of Damgård et al. [8]. Both these constructions rely on projective almost-universal hash-proof system (HPS) as a common building block, and we observe that on a high level, our entropic post-challenge BLT security relies on the *statistical soundness* property of the HPS. However, we choose to start with the construction of Damgård et al. [8] due to its simplicity.

Entropic Post-challenge IND-CCA-BLT PKE: Next, we show how to upgrade the *entropic restricted* post-challenge IND-CCA-BLT security to *entropic* post-challenge IND-CCA-BLT security. In the entropic notion, the adversary can query arbitrary ciphertexts to the (tampered) decryption oracles, as opposed to the entropic restricted notion, where the adversary can only query well-formed (valid) ciphertexts to the oracle. The adversary also has access to the normal (non-tampered) decryption oracle $\mathsf{Dec}(sk, \cdot)$ both in the pre- and post-challenge phase as in the IND-CCA security game. The transformation follows the classical paradigm of converting a CPA-secure PKE to a CCA-secure one by appending to the ciphertext a zero knowledge argument proving the knowledge of the plaintext. Similar transformation was shown in [8] for converting a restricted IND-CCA-BLT secure PKE scheme to a full fledged IND-CCA-BLT secure PKE scheme in the context of pre-challenge leakage and tampering. We observe that the same transformation goes through in the context of post-challenge leakage and tampering as well, and also when the PKE scheme is entropic.

Upgrading to Full Fledged (Non-entropic) Security: We then show how to compile such an *entropic* post-challenge IND-CCA-BLT secure PKE scheme to a *full-fledged* post-challenge IND-CCA-BLT secure PKE scheme. For this, we resort to our split-state leakage and tampering restriction[2]. On a high level, our construction bears similarity with the construction of [16], although the PKE scheme of [16] was only proven to be CPA secure against leakage attacks. We appropriately modify their construction to prove our scheme to be CCA-secure and resilient to joint leakage and tampering attacks. To make the construction more modular, we first show how to construct post-challenge IND-CCA-BLT secure key encapsulation mechanism (KEM) and later show how to compile it to a full-fledged PKE scheme.

On a high level, to generate an encapsulated symmetric key, we generate a key pair (vk, sk) of a strong one-time signature (OTS) scheme. We then use two instances of the entropic scheme to encrypt two random strings x_1 and x_2 independently, with the verification key vk as the label/tag to generate two ciphertexts c_1 and c_2 respectively. The ciphertext $c = (c_1, c_2)$ is then signed using the OTS scheme to generate a signature, say, σ. Finally, we apply a seedless 2-source extractor to both x_1 and x_2 to generate the encapsulated key. We then output the final ciphertext $c = (vk, c_1, c_2, \sigma)$. On a high level, the security of the entropic scheme guarantees that both the strings x_1 and x_2 still retain enough average min-entropy even after chosen-ciphertext leakage and tampering attacks (even in the post-challenge phase). In addition, the split-state model ensures that the strings are independent. At this point, we can use an average-case seedless 2-source extractor to extract a random encapsulation key from both the strings. The trick of generating a key pair of an OTS and setting the verification key vk as a tag/label while encrypting, ensures that, a tag cannot be re-used by an adversary in a decryption or tampering query, hence preventing "mix-and-match" attacks (In fact, to re-use that tag, the adversary essentially has to forge a signature under vk).

Compiling to a Post-challenge IND-CCA-BLT PKE: Finally, we show how to construct a IND-CCA-BLT secure PKE from a IND-CCA-BLT secure KEM as above. One natural idea to achieve this is to use standard hybrid encryption technique, where a symmetric-key encryption (SKE) scheme is used to encrypt the message using the derived encapsulation key. However, we point out, that unlike in standard PKE or even in leakage-resilient PKE settings, this transformation needs a little careful analysis in the context of tampering. This is because the adversary can also ask decryption queries with respect to the tampered keys, and the security of the challenge ciphertext should hold even given these tampered decryption oracle responses. This is not directly guaranteed by standard hybrid encryption paradigm. However, we leverage on the security guarantee of our KEM scheme and show that it is indeed possible to argue the above security. In particular, our KEM scheme guarantees that the average min-entropy of the challenge KEM key K^* is negligibly close to an uniform distribution over the KEM key space, even given many tampered keys $K = (\widetilde{K}_1, \cdots, \widetilde{K}_t)$. So, in the hybrid, we can replace the key K^* with a uniform random key. This implies that, with very high probability, K^* is independent

[2] For our construction the secret key is split into only *two* parts/splits, which is the optimal.

of the tampered key distribution, and hence any function of the tampered keys (in particular decryption function). We can then rely on the (standard) CCA security of the SKE to argue indistinguishability of the challenge messages. Finally, combining all the above ideas together, we obtain the full construction of a post-challenge IND-CCA-BLT secure PKE scheme, thus solving the open problem posed by Faonio and Venturi [12] (Asiacrypt 2016).

Lastly, we note that, it is instructive to compare our approach of constructing post-challenge leakage and tamper-resilient PKE construction with that of Liu and Lysyanskaya [19]. We observe that the framework of [19] instantiated with a non-malleable extractor, would already produce a scheme with security against post-challenge tampering. However, their model is not comparable with ours in the following sense. In particular, the framework of [19] considers securing any (deterministic) cryptographic functionality against leakage and tampering attacks, where the leakage and tampering functions apply only on the memory of the device implementing the functionality, and not on its computation. This is because the construction of [19] relies on a (computationally secure) leakage-resilient non-malleable code, which allow only leakage and tampering on the memory of the device. However, in our model, we allow the adversary to leak from the memory and also allow to tamper with the internal computations (modeled by giving the adversary access to tampered decryption oracles). In this sense, our model is more general, as it also considers tampering with the computation. However, a significant feature of the framework of [19] is that, it considers the model of continual leakage and tampering (in split-state), whereas our model considers bounded leakage and tampering (as in [8]) in split-state.

1.2 Organization

The rest of the paper is organized as follows. In Sect. 2, we provide the necessary preliminaries required for our constructions. In Sect. 3, we give our definition of entropic post-challenge IND-CCA-BLT secure PKE schemes and its restricted notion. In Sect. 3.2, we show our construction of entropic restricted post-challenge IND-CCA-BLT secure PKE and show the transformation from the entropic restricted notion to the entropic notion in Sect. 3.3. In Sect. 4, we present the security definition of post-challenge IND-CCA-BLT secure KEM scheme and show a generic compiler from entropic post-challenge IND-CCA-BLT secure PKE scheme to a post-challenge IND-CCA-BLT secure PKE scheme in the standard model. Section 5 shows the generic transformation from such a KEM scheme to a full fledged IND-CCA-BLT secure PKE scheme secure against post-challenge leakage and tampering attacks. Finally Sect. 6 concludes the paper.

2 Preliminaries

2.1 Notations

For $n \in \mathbb{N}$, we write $[n] = \{1, 2, \cdots, n\}$. If x is a string, we denote $|x|$ as the length of x. For a set $\mathcal{X}$, we write $x \overset{\$}{\leftarrow} \mathcal{X}$ to denote that element x is chosen uniformly at random from $\mathcal{X}$. For a distribution or random variable X, we denote $x \leftarrow X$ the action of sampling an element x according to X. When A is an algorithm, we write $y \leftarrow A(x)$ to denote a run of A on input x and output y; if A is randomized, then y is a random variable and $A(x;r)$ denotes a run of A on input x and randomness r. An algorithm A is probabilistic polynomial-time (PPT) if A is randomized and for any input $x, r \in \{0,1\}^*$; the computation of $A(x;r)$ terminates in at most $poly(|x|)$ steps. For a set S, we let U_S denote the uniform distribution over S. For an integer $\alpha \in \mathbb{N}$, let U_α denote the uniform distribution over $\{0,1\}^\alpha$, the bit strings of length α. Throughout this paper, we denote the security parameter by κ. Vectors are written in boldface. Given a vector $\mathbf{x} = \{x_1, \cdots, x_n\}$, and some integer a, we write $a^{\mathbf{x}}$ to denote the vector $(a^{x_1}, \cdots, a^{x_n})$. Let D_1 and D_2 be two distributions on a finite set $\mathcal{S}$. We denote by $\left|D_1 - D_2\right|$ the statistical distance between them. For random variables X, Y, we denote min-entropy (conditional min-entropy) of X as $\mathrm{H}_\infty(X)$ ($\widetilde{\mathrm{H}}_\infty(X|Y)$) respectively. We assume that the reader is familiar with the results related to (conditional) min- entropy, and we refer to the full version of our paper [6] for these definitions. We denote a distribution supported on $\{0,1\}^n$ with min-entropy k to be an (n,k)-source.

2.2 Two Source Extractors

In this section, we give an overview of two-source extractors [7,21,22] and their generalization, which will be required for our work.

Definition 1 (Seedless 2-source Extractor). *A function* $\mathsf{Ext2} : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^m$ *is a seedless 2-source extractor at min-entropy* k *and error* ϵ *if it satisfies the following property: If* X *and* Y *are independent* (n,k)*-sources, it holds that* $\left| \ (\mathsf{Ext2}(X,Y) - U_m) \ \right| < \epsilon$. *where* U_m *refer to a uniform* m*-bit string.*

Definition 2 (Average-case Seedless 2-source Extractor). *A function* $\mathsf{Ext2} : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^m$ *is an average-case seedless 2-source extractor at min-entropy* k *and error* ϵ *if it satisfies the following property: If for all random variables* $X, Y \in \{0,1\}^n$ *and* Z*, such that, conditioned on* Z*,* X *and* Y *are independent* (n,k)*-sources, it holds that* $\left| \ ((\mathsf{Ext2}(X,Y), Z) - (U_m, Z)) \ \right| < \epsilon$.

Lemma 1 [16]**.** *For any* $\delta > 0$*, if* $\mathsf{Ext2} : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^m$ *is a (worst-case)* $(k - \log \frac{1}{\delta}, \epsilon)$*-2-source extractor, then* $\mathsf{Ext2}$ *is an average-case* $(k, \epsilon + 2\delta)$*-2-source extractor.*

2.3 True Simulation Extractable Non-interactive Zero Knowledge Argument System

In our construction, we require the notion of (same-string) *true-simulation extractable non-interactive zero knowledge argument system* (tSE-NIZK) first introduced in [9] and also its extension to support labels/tags. This notion is similar to the notion of simulation-sound extractable NIZKs [14] with the difference that the adversary has oracle access to *simulated* proofs only for *true* statements, in contrast to any arbitrary statement as in simulation-sound extractable NIZK argument system. In particular, we require the standard properties of *completeness*, *soundness* and *composable zero-knowledge*. Additionally, we also require the existence of another PPT *extractor* Ext which extracts a valid witness from any proof produced by a malicious prover $\mathcal{P}^*$, even if $\mathcal{P}^*$ has previously seen some *simulated proofs* for *true* statements. We refer the reader to the full version of our paper [6] for the formal definition of tSE-NIZK. For our purpose, it is sufficient to rely on the (weaker) notion of *one-time* strong true simulation extractability, where the adversary can query the simulation oracle $\mathcal{SIM}_{\mathsf{tk}}(.)$ *only once*. Dodis et al. [9] showed how to generically construct tSE-NIZK argument systems supporting labels starting from any (labeled) CCA-secure PKE scheme and a (standard) NIZK argument system.

3 Entropic Post-challenge IND-CCA-BLT Secure PKE

In this section, we introduce the definition of *entropic post-challenge* IND-CCA-secure PKE resilient to both pre- and post-challenge bounded leakage and tampering (BLT) attacks. In Sect. 3.1, we define a relaxation of our entropic notion, which we call *entropic restricted post-challenge* IND-CCA BLT secure PKE. We show that a variant of the cryptosystem of Boneh et al. [4] with appropriate parameters, satisfies our entropic restricted notion of security (see Sect. 3.2). Finally, in Sect. 3.3, we show a generic transformation from our entropic restricted notion to the full-fledged entropic post-challenge IND-CCA-BLT secure PKE scheme. Before defining these notions, we explain the working of the leakage oracle and the tampering oracle.

The Leakage Oracle. In order to model *key leakage* attacks, we assume that the adversary may access a leakage oracle $O^{\lambda}_{sk}(.)$, subject to some restrictions. The adversary can query this oracle with arbitrary efficiently computable (poly-time) leakage functions f and receive $f(sk)$ in response, where sk denotes the secret key. The restriction is that the output length of f must be less than $|sk|$. Specifically, following the works of [2,9], we require the output length of the leakage function f to be at most λ bits, which means the entropy loss of sk is at most λ bits upon observing $f(sk)$. Formally, we define the bounded leakage function family $\mathcal{F}_{bbd}(\kappa)$. The family $\mathcal{F}_{bbd}(\kappa)$ is defined as the class of all polynomial-time computable functions: $f : \{0,1\}^{|sk|} \rightarrow \{0,1\}^{\lambda}$, where $\lambda < |sk|$. We then require that the leakage function submitted by the adversary should satisfy that $f \in \mathcal{F}_{bbd}(\kappa)$.

The Tampering Oracle. To model related key attacks, the adversary is given access to a tampering oracle. Let $\mathcal{T}_{SK}$ denote the class of functions from SK to SK, where SK is the secret key space. The adversary may query the tampering oracle with arbitrary functions of its choice from $\mathcal{T}_{SK}$ and the number of such queries is *bounded* (say $t \in \mathbb{N}$). In the i^{th} tampering query ($i \in [t]$), the adversary chooses a function $T_i \in \mathcal{T}_{SK}$ and gets access to the (tampered) decryption oracle $\mathsf{Dec}(\widetilde{sk}_i, \cdot)$, where $\widetilde{sk}_i = T_i(sk)$. The adversary may ask polynomially many decryption queries with respect to the tampered secret key $\widetilde{sk}_i$. In other words, the adversary gets access to information through decryption oracle executed on keys related to the original secret key, where the relations are induced by the tampering functions. If the encryption scheme supports labels, i.e., it is a labeled encryption scheme, the adversary gets access to the (tampered) decryption oracle $\mathsf{Dec}(\widetilde{sk}_i, \cdot, \cdot)$, where the third coordinate is a placeholder for labels. Also, the adversary gets access to the (tampered) decryption oracle both in the pre- and post-challenge phases. Another (obvious) restriction that is imposed on the tampering functions is that: In the post-challenge phase, when the adversary gets access to the (tampered) decryption oracles with respect to the challenge ciphertext c^*, it should be the case that $T_i(sk) \neq sk$, i.e., the post-challenge tampering functions T_i should not be identity functions with respect to the challenge ciphertext[3].

Definition 3 (Entropic Post-challenge IND-CCA-BLT Secure PKE). Our definition of entropic post-challenge IND-CCA-BLT secure PKE can be seen as an *entropic version* of the notion of IND-CCA-BLT secure PKE introduced in [8], augmented with post challenge leakage and tampering queries. Informally, our definition captures the intuition that if we start with a message M with high min-entropy, the message M still *looks random* to an adversary who gets to see the ciphertext, some leakage information (even if this leakage happens after observing the ciphertext), and access to the tampering oracle (both in pre- and post-challenge phase) as defined above.

Formally, we define two games- "real" game and a "simulated" game. For simplicity, we assume the message is chosen from U_k, i.e, the uniform distribution over k bit strings. In general, it can be chosen from any arbitrary distribution as long as the message has min-entropy k. Let $(\lambda_{\mathsf{pre}}, \lambda_{\mathsf{post}})$ and $(t_{\mathsf{pre}}, t_{\mathsf{post}})$ denote the leakage bounds and the number of tampering queries allowed in the pre- and post-challenge phases respectively.

The "real" game. Given the parameters $\big(k, (\lambda_{\mathsf{pre}}, \lambda_{\mathsf{post}}), (t_{\mathsf{pre}}, t_{\mathsf{post}})\big)$ and a labeled encryption scheme $\mathsf{E\text{-}BLT} = (\mathsf{E\text{-}BLT.SetUp}, \mathsf{E\text{-}BLT.Gen}, \mathsf{E\text{-}BLT.Enc}, \mathsf{E\text{-}BLT.Dec})$, the real game is defined as follows:

0. **Sampling:** *The challenger chooses a random message* $m \xleftarrow{\$} U_k$.
1. **SetUp:** *The challenger runs params* $\leftarrow \mathsf{E\text{-}BLT.SetUp}(1^\kappa)$ *and sends params to the adversary* $\mathcal{A}$. *The public parameters params are taken as (implicit) input by all other algorithms.*
2. **Key Generation:** *The challenger chooses* $(sk, pk) \leftarrow \mathsf{E\text{-}BLT.Gen}(params)$ *and sends pk to* $\mathcal{A}$. *Set* $L^{\mathsf{pre}} = L^{\mathsf{post}} = 0$.

[3] When $T_i(sk) = sk$, and the adversary gets access to the tampering oracle with respect to c^*, it is emulating the scenario when it gets decryption oracle access with respect to sk on c^*, which is anyway disallowed in the IND-CCA-2 security game.

3. **Pre-challenge Leakage:** *In this phase, the adversary $\mathcal{A}$ makes a pre-challenge leakage query, specifying a function $f_{\mathsf{pre}}(.)$. If $L^{\mathsf{pre}} + |f_{\mathsf{pre}}(sk)| \leq \lambda_{\mathsf{pre}}$, then the challenger replies with $f_{\mathsf{pre}}(sk)$, and sets $L^{\mathsf{pre}} = L^{\mathsf{pre}} + |f_{\mathsf{pre}}(sk)|$. Otherwise, it ignores this query.*
4. **Pre-challenge Tampering queries:** *The adversary $\mathcal{A}$ may adaptively ask at most t_{pre} number of pre-challenge tampering queries. In the i^{th} tampering query ($i \in [t_{\mathsf{pre}}]$), the adversary chooses $T_i \in \mathcal{T}_{SK}$, and gets access to the decryption oracle* $\mathsf{E\text{-}BLT.Dec}(\widetilde{sk}_\theta, \cdot, \cdot)$[4] *(where $1 \leq \theta \leq i$). In other words, the decryption oracle may be queried with any of the tampered keys obtained till this point. We assume that, the total number of decryption oracle queries be $q(k)$, for some polynomial $q(k)$. Note that, when $T_\theta(sk) = sk$, $\mathcal{A}$ gets access to the (normal) decryption oracle.*
5. **Challenge:** *In this phase, the adversary submits a label* (as a bit-string) L^*. *The challenger encrypts the message m chosen at the beginning of the game as $c^* \leftarrow \mathsf{E\text{-}BLT.Enc}(pk, m, L^*)$ and sends c^* to $\mathcal{A}$.*
6. **Post-challenge Leakage:** *In this phase, the adversary $\mathcal{A}$ makes a post-challenge leakage query, specifying a function $f_{\mathsf{post}}(.)$. If $L^{\mathsf{post}} + |f_{\mathsf{post}}(sk)| \leq \lambda_{\mathsf{post}}$, then the challenger replies with $f_{\mathsf{post}}(sk)$, and sets $L^{\mathsf{post}} = L^{\mathsf{post}} + |f_{\mathsf{post}}(sk)|$. Otherwise, it ignores this query.*
7. **Post-challenge Tampering queries:** *The adversary $\mathcal{A}$ may adaptively ask t_{post} number of post-challenge tampering queries. In the j^{th} tampering query ($j \in [t_{\mathsf{post}}]$), the adversary chooses $T_j \in \mathcal{T}_{sk}$, and gets access to the decryption oracle $\mathsf{E\text{-}BLT.Dec}(\widetilde{sk}_\rho, \cdot, \cdot)$ ($1 \leq \rho \leq j$). We assume that, the total number of decryption oracle queries be $q'(k)$, for some polynomial $q'(k)$. However, here we impose the restriction that: $\mathcal{A}$ is not allowed to query the pair (c^*, L^*) to the (tampered) decryption oracle(s) $\mathsf{E\text{-}BLT.Dec}(\widetilde{sk}_\rho, \cdot, \cdot)$.*

Note that all these queries can be made *arbitrarily* and *adaptively* in nature. We denote the message m chosen at the onset of this game as M^{rl} to emphasize that it is used in the real game. Let the sets Q_{pre} and Q_{post} contain the tuples of the form $\left\{(\widetilde{m}_{i_1}, (c_{i_1}, L_{i_1})), \cdots, (\widetilde{m}_{i_{q(\kappa)}}, (c_{i_{q(\kappa)}}, L_{i_{q(\kappa)}}))\right\}_{i=1}^{t_{\mathsf{pre}}}$ and $\left\{(\widetilde{m}_{j_1}, (c_{j_1}, L_{j_1})), \cdots, (\widetilde{m}_{j_{q(\kappa)}}, (c_{i_{q'(\kappa)}}, L_{i_{q'(\kappa)}}))\right\}_{j=1}^{t_{\mathsf{post}}}$ respectively, for some polynomials $q(\kappa)$ and $q'(\kappa)$. Let $\mathcal{L}_{\mathsf{pre}}$ and $\mathcal{L}_{\mathsf{post}}$ be the random variables corresponding to the pre- and post-challenge leakages. We define the view of the adversary $\mathcal{A}$ in the real game as $\mathsf{View}^{\mathsf{rl}}_{\mathsf{E\text{-}BLT},\mathcal{A}}(\kappa) = (\mathsf{rand}, \mathcal{L}_{\mathsf{pre}}, Q_{\mathsf{pre}}, c^*, \mathcal{L}_{\mathsf{post}}, Q_{\mathsf{post}})$, where rand denotes the random coins used by the adversary in the game. Finally, we denote by $(M^{\mathsf{rl}}, \mathsf{View}^{\mathsf{rl}}_{\mathsf{E\text{-}BLT},\mathcal{A}})$ the joint distribution of the message M^{rl} and $\mathcal{A}$'s view in a real game with M^{rl}.

The "simulated" game: In the simulated game, we replace the challenger from above by a simulator Simu that interacts with $\mathcal{A}$ in any way that it sees fit.

[4] Recall when we write $\mathsf{Dec}(\widetilde{sk}_\theta, \cdot, \cdot)$, the second coordinate is the placeholder for ciphertexts input by the adversary; whereas the third coordinate is the placeholder for labels.

Simu gets a uniformly chosen message M^{sm} as input and it has to simulate the interaction with $\mathcal{A}$ conditioned on M^{sm}. We denote the view of the adversary in the simulated game by $\mathsf{View}^{\mathsf{sm}}_{\mathsf{Simu},\mathcal{A}}(\kappa) = (\mathsf{rand}^{\mathsf{sm}}, \mathcal{L}^{\mathsf{sm}}_{\mathsf{pre}}, Q^{\mathsf{sm}}_{\mathsf{pre}}, c^{\mathsf{sm}}, \mathcal{L}^{\mathsf{sm}}_{\mathsf{post}}, Q^{\mathsf{sm}}_{\mathsf{post}})$. Now, we define what it means for the encryption scheme $\mathsf{ER\text{-}BLT}$ to be entropic restricted post-challenge (bounded) leakage and tamper-resilient.

Definition 4 (Entropic restricted post-challenge IND-CCA-BLT security). *Let $\big(k, (\lambda_{\mathsf{pre}}, \lambda_{\mathsf{post}}), (t_{\mathsf{pre}}, t_{\mathsf{post}})\big)$ be parameters as stated above, let $\mathcal{T}_{SK}$ be the family of allowable tampering functions. A public key encryption scheme is said to be entropic restricted post-challenge IND-CCA-BLT secure with respect to all these parameters if there exists a simulator* Simu*, such that, for every PPT adversary $\mathcal{A}$ the following two conditions hold:*

1. $(M^{\mathsf{rl}}, \mathsf{View}^{\mathsf{rl}}_{\mathsf{E\text{-}BLT},\mathcal{A}}(\kappa)) \approx_c (M^{sm}, \mathsf{View}^{\mathsf{sm}}_{\mathsf{Simu},\mathcal{A}}(\kappa))$*, i.e, the above two ensembles (indexed by the security parameter) are computationally indistinguishable.*
2. *The average min-entropy of the message M^{sm} given $\mathsf{View}^{\mathsf{sm}}_{\mathsf{Simu},\mathcal{A}}(\kappa)$ is*

$$\widetilde{\mathrm{H}}_{\infty}(M^{\mathsf{sm}} \mid \mathsf{View}^{\mathsf{sm}}_{\mathsf{Simu},\mathcal{A}}(\kappa)) \geq k - \lambda_{\mathsf{post}} - \mathcal{F}(t_{\mathsf{post}}).$$

where $\mathcal{F}(t_{\mathsf{post}})$ denotes the entropy loss due to post-challenge tampering queries, and the tampering functions come from the class $\mathcal{T}_{SK}$.[5]

Intuitively, even after the adversary sees the encryption of the message, pre- and post-challenge leakages and the output of the (tampered) decryption oracle both in the pre- and post-challenge phase, the message M^{sm} still retains its initial entropy, except for the entropy loss due to post-challenge leakage and tampering.

3.1 Entropic Restricted Post-challenge IND-CCA-BLT Secure PKE

We now define the notion of *entropic restricted* post-challenge IND-CCA-BLT secure PKE (denoted by $\mathsf{ER\text{-}BLT}$), which is a relaxation of the notion of the entropic post-challenge IND-CCA-BLT secure PKE. The difference between the two notions is with respect to the working of (tampered) decryption oracle, as defined in the real game in Definition 3. In particular, in our entropic restricted notion of security, the adversary *cannot* make pre- and post-challenge decryption queries with respect to the original secret key (unlike the entropic notion in Sect. 3) and working of the (tampered) decryption oracle is *modified* as follows:

Modified Decryption Oracle: In the restricted post-challenge IND-CCA-BLT security game, the adversary is not given full access to the tampering oracle. Instead, the adversary is allowed to see the output of the (tampered) decryption oracle for only those ciphertexts c, for which he already knows the plaintext m and the randomness r used to encrypt it (using the original

[5] In our construction, we will show that $\mathcal{F}(t_{\mathsf{post}}) = t_{\mathsf{post}} \log p$, i.e., for each post-challenge tampering query we have to leak *only* one element of the base group $\mathbb{G}$ of prime order p. This single element is sufficient to simulate polynomially many (modified) decryption queries with respect to each tampering query.

public key). This restricts the power of the adversary to submit only "*well-formed*" ciphertexts to the tampering oracle. In particular, in the i^{th} tampering query the adversary chooses a function $T_i \in \mathcal{T}_{SK}$ and gets access to a (modified) decryption oracle $\mathsf{ER\text{-}BLT.Dec}^*(\widetilde{sk}_i, \cdot, \cdot)$, where $\widetilde{sk}_i = T_i(sk)$. This oracle answers polynomially many queries of the following form: Upon input a pair $(m, r) \in \mathcal{M} \times \mathcal{R}$, (where $\mathcal{M}$ and $\mathcal{R}$ are the message space and randomness space of the PKE respectively), compute $c \leftarrow \mathsf{ER\text{-}BLT.Enc}(pk, m; r)$ and output a plaintext $\widetilde{m} = \mathsf{ER\text{-}BLT.Dec}(\widetilde{sk}_i, c)$ under the current tampered key.
The real and simulated game for the above entropic restricted post-challenge IND-CCA-BLT game, apart from the above restrictions, is identical to the real and simulated games of the entropic post-challenge IND-CCA-BLT secure PKE as defined in Definition 3. In particular, using the same notations from Definition 3, we denote the view of the adversary in the entropic restricted game as $\mathsf{View}^{\mathsf{rl}}_{\mathsf{ER\text{-}BLT},\mathcal{A}}(\kappa) = (\mathsf{rand}, \mathcal{L}_{\mathsf{pre}}, Q_{\mathsf{pre}}, c^*, \mathcal{L}_{\mathsf{post}}, Q_{\mathsf{post}})$, where Q_{pre} and Q_{post} contain answers to the (tampered) decryption oracle queries as described above with respect to the tampered secret keys.

3.2 Construction of Entropic Restricted Post-challenge IND-CCA-BLT Secure PKE

In this section, we show how to construct a CCA-2 secure entropic restricted post-challenge PKE secure against bounded leakage and tampering (BLT) attacks. We show that a variant of the encryption scheme proposed by Boneh et al. (referred to as BHHO cryptosystem from herein) [4] is entropic restricted post-challenge IND-CCA-BLT secure. It was shown in [8] that the (modified) BHHO cryptosystem is a restricted (pre-challenge) IND-CCA-BLT secure PKE. However, we observe that the same variant of the BHHO cryptosystem with the parameters appropriately modified satisfies our new notion of entropic security, even when the adversary is given post-challenge leakage and access to (restricted) tampering oracle (even in the post-challenge phase).

– $\mathsf{ER\text{-}BLT.SetUp}(1^\kappa)$: Choose a group $\mathbb{G}$ of prime order p with generator g. Set $params := (\mathbb{G}, g, p)$. All the algorithms take $params$ as implicit input.
– $\mathsf{ER\text{-}BLT.Gen}(params)$: Sample random vectors $\mathbf{x}$, $\boldsymbol{\alpha} \in \mathbb{Z}_p^\ell$; compute $g^{\boldsymbol{\alpha}} = (g_1, \cdots, g_\ell)$, and $h = \prod_{i=1}^{\ell} g_i^{x_i}$. Set $sk := \mathbf{x} = (x_1, \cdots, x_\ell)$ and $pk := (h, g^{\boldsymbol{\alpha}})$
– $\mathsf{ER\text{-}BLT.Enc}(pk, m)$: Sample $r \leftarrow \mathbb{Z}_p$, and return $c := (g_1^r, \cdots, g_\ell^r, h^r \cdot m)$
– $\mathsf{ER\text{-}BLT.Dec}(sk, c)$: Parse c as $(c_1, \cdots, c_\ell, d)$ as sk as $(x_1, \cdots, x_\ell)$., and outputs $m \leftarrow d / \prod_{i=1}^{\ell} (g_i^r)^{x_i}$

It is easy to verify the correctness of the above cryptosystem.

Theorem 1. *Let $\kappa \in \mathbb{N}$ be the security parameter, and assume that the* DDH *assumption holds in group $\mathbb{G}$. The* BHHO *cryptosystem is entropic restricted post-challenge* IND-CCA-$\big(k, (\lambda_{\mathsf{pre}}, \lambda_{\mathsf{post}}), (t_{\mathsf{pre}}, t_{\mathsf{post}})\big)$-BLT *secure, where*

$$\lambda_{\mathsf{pre}} + \lambda_{\mathsf{post}} \leq \big(\ell - 2 - t_{\mathsf{pre}} - t_{\mathsf{post}}\big) \log p - \omega(\log \kappa) \quad \textit{and} \quad (t_{\mathsf{pre}} + t_{\mathsf{post}}) \leq \ell - 3.$$

Proof. Before proceeding with the proof of the above theorem, we prove a lemma (Lemma 2) that essentially shows that the BHHO cryptosystem is entropic leakage-resilient with respect to pre- and post-challenge leakage, i.e., it satisfies the notion of entropic restricted post-challenge IND-CCA-$\big(k, (\lambda'_{\mathsf{pre}}, \lambda'_{\mathsf{post}}), (0, 0)\big)$-BLT security (the adversary has no access to the tampering oracle), for appropriate choice of parameters. We then prove the above theorem by using Lemma 2 and showing a leakage to tamper reduction to take care of pre- and post-challenge tampering queries.

Lemma 2. *The* BHHO *cryptosystem described above is entropic restricted post-challenge* IND-CCA-$\big(k, (\lambda'_{\mathsf{pre}}, \lambda'_{\mathsf{post}}), (0, 0)\big)$-BLT *secure, where*

$$\lambda'_{\mathsf{pre}} + \lambda'_{\mathsf{post}} \leq \big(\ell - 2\big) \log p - \omega(\log \kappa)$$

Proof. To prove Lemma 2 we need to describe a simulator, whose answers to the adversary are indistinguishable from the real game, and at the same time leave enough min-entropy in the message m. The main idea of the proof follows from the observation that the BHHO cryptosystem can be viewed as a *hash proof system* (HPS) (see [6] for the definition of HPS), with DDH-like tuples as valid ciphertexts, and non-DDH tuples as invalid ciphertexts. In the real game, the challenger samples a valid ciphertext (along with a witness) and proceeds as in the original construction, whereas in the simulated game a random invalid ciphertext is sampled. The indistinguishability of the real and simulated games is implied by the subset membership problem. The left-over hash lemma then guarantees uniformity of the challenge message. For details of the proof, please refer to the full version of our paper [6].

We now proceed to prove our main theorem. Let us assume that there exists an adversary $\mathcal{A}$ that breaks the entropic restricted post-challenge IND-CCA $\big(k, (\lambda_{\mathsf{pre}}, \lambda_{\mathsf{post}}), (t_{\mathsf{pre}}, t_{\mathsf{post}})\big)$-BLT security with non-negligible advantage. We construct an adversary $\mathcal{A}'$ against the entropic restricted post-challenge IND-CCA $\big(k, (\lambda'_{\mathsf{pre}}, \lambda'_{\mathsf{post}}), (0, 0)\big)$-BLT security, with the same advantage. The main idea behind this proof is *leakage to tamper reduction*. For each tampering query made by the adversary, the reduction simply leaks a single group element from $\mathbb{Z}_p$, and simulates polynomially many decryption queries under that tampered key using the leaked element. Hence, the reduction has to leak $(t_{\mathsf{pre}} + t_{\mathsf{post}}) \log p$ bits in all. We appropriately set the parameters of BHHO to ensure that the message still has enough min-entropy, even given the responses of the tampering oracle. We refer the reader to the full version [6] for the detailed proof.

3.3 The General Transformation

In this section, we show a general transformation from an entropic-restricted post-challenge IND-CCA-BLT secure PKE to an entropic post-challenge IND-CCA-BLT secure PKE scheme (see Fig. 1). Let $\mathsf{ER\text{-}BLT} = (\mathsf{ER\text{-}BLT.SetUp}, \mathsf{ER\text{-}BLT.Gen}, \mathsf{ER\text{-}BLT.Enc}, \mathsf{ER\text{-}BLT.Dec})$ be an entropic restricted post-challenge IND-CCA-$\big(k, (\lambda_{\mathsf{pre}}, \lambda_{\mathsf{post}}), (t_{\mathsf{pre}}, t_{\mathsf{post}})\big)$-BLT secure PKE scheme, and let

$\Pi = (\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ be a one-time strong tSE-NIZK argument system supporting labels for the following relation:

$$\mathbb{R}_{\mathsf{ER\text{-}BLT}} = \{(m, r), (pk, c) \mid c = \mathsf{ER\text{-}BLT.Enc}(pk, m; r)\}$$

Let $\mathsf{E\text{-}BLT} = (\mathsf{E\text{-}BLT.SetUp}', \mathsf{E\text{-}BLT.Gen}', \mathsf{E\text{-}BLT.Enc}', \mathsf{E\text{-}BLT.Dec}')$ be an entropic post-challenge IND-CCA-BLT secure PKE.

Theorem 2. *Let* $\mathsf{ER\text{-}BLT}$ *be an entropic-restricted post-challenge* IND-CCA-$(k, (\lambda_{\mathsf{pre}}, \lambda_{\mathsf{post}}), (t_{\mathsf{pre}}, t_{\mathsf{post}}))$-BLT *secure PKE scheme,* Π *be a one-time strong tSE NIZK argument system supporting label for the relation* $\mathbb{R}_{\mathsf{ER\text{-}BLT}}$*, then the above encryption scheme* $\mathsf{E\text{-}BLT}$ *is an entropic post-challenge* IND-CCA-$(k, (\lambda_{\mathsf{pre}}, \lambda_{\mathsf{post}}), (t_{\mathsf{pre}}, t_{\mathsf{post}}))$-BLT *secure PKE scheme.*

Define the encryption scheme $\mathsf{E\text{-}BLT}$ as follows:

1. $\mathsf{E\text{-}BLT.SetUp}'(1^\kappa)$: Obtain $params \leftarrow \mathsf{ER\text{-}BLT.SetUp}(1^\kappa)$, and sample $(\mathsf{crs}, \mathsf{tk}, \mathsf{ek}) \leftarrow \mathsf{Gen}(1^\kappa)$. Set $params' := (params, \mathsf{crs})$
2. $\mathsf{E\text{-}BLT.Gen}'(params')$: Obtain $(pk, sk) \leftarrow \mathsf{ER\text{-}BLT.Gen}(params)$; set $pk' = pk$, and $sk' = sk$.
3. $\mathsf{E\text{-}BLT.Enc}'(pk, m, \mathsf{L})$: On input the public key pk, a message $m \in \mathcal{M}$ and a label L , sample $r \overset{\$}{\leftarrow} \mathcal{R}$, and compute $c \leftarrow \mathsf{ER\text{-}BLT.Enc}(pk, m; r)$, $\pi \leftarrow \mathsf{P}(\mathsf{crs}, \mathsf{L}, (m, r), (pk, c))$. Output $c' = (c, \pi)$
4. $\mathsf{E\text{-}BLT.Dec}'(sk, c', \mathsf{L})$: Parse c' as $c' = (c, \pi)$. Check if $\mathsf{V}(\mathsf{crs}, \mathsf{L}, (pk, c), \pi) = 1$. If not output $\perp$, else output $m = \mathsf{ER\text{-}BLT.Dec}(sk, c)$

Fig. 1. Entropic post-challenge IND-CCA-BLT PKE scheme $\mathsf{E\text{-}BLT}$

Proof Sketch. We now give an intuitive proof sketch of the above theorem. Informally, the zero-knowledge argument enforces the adversary to submit to the (tampered) decryption oracle only *valid* ciphertexts, for which he knows the corresponding plaintext (and the randomness used to encrypt it). The plaintext-randomness pair (m, r) (which acts as a witness) can then be extracted using the extraction trapdoor of the tSE-NIZK argument system, thus allowing to reduce entropic IND-CCA BLT security to entropic restricted IND-CCA BLT security. Since the extraction trapdoor is never used in the real encryption scheme, the adversary neither gets any leakage from it, nor gets to tamper with it. This essentially makes the (tampered) decryption oracle useless and the adversary learns *no* additional information from the decryption oracle access. The proof also relies on the fact that the CRS is untamperable, a notion that is used in all the previous works [8,12]. This can be achieved by (say) hard-coding the CRS in the encryption algorithm. The detailed proof of this theorem can be found in the full version [6] of our paper.

4 Post-challenge IND-CCA-BLT Secure KEM in Split-State Model

In this section, we present our construction of post-challenge IND-CCA-BLT secure Key Encapsulation Mechanism (KEM) in the (bounded) split-state leakage and tampering model. Note that, achieving security against post-challenge leakage and tampering in its most general form is impossible as already shown in [8,16,20], even if a single bit of leakage is allowed or the adversary is allowed to ask even a single tampering query after receiving the challenge ciphertext. To this end, we resort to the 2-split-state leakage and tampering model. In this model, the secret key of the KEM scheme is split into two disjoint parts, and the adversary can ask arbitrary (pre- and post-challenge) leakage and tampering queries on each of these two parts *independently*. However, the adversary is allowed to adaptively ask leakage/tampering functions depending on the answers of the previous queries. The tampering queries allow the adversary to have access to the tampered decryption oracle. The adversary also gets access to the (standard) decryption oracle by specifying the tampering functions to be identity functions. Finally, the adversary has to guess whether the challenger KEM key is a randomly sampled key or a real key. Due to space constraints, we refer the reader to the full version [6] for the formal definition and the security model for IND-CCA-BLT secure KEM.

4.1 Construction of Post-challenge IND-CCA-BLT Secure KEM

We now show the construction of our post-challenge/after-the-fact IND-CCA-BLT secure KEM scheme $\mathcal{KEM} = (\mathcal{KEM}.\mathsf{Setup}, \mathcal{KEM}.\mathsf{Gen}, \mathcal{KEM}.\mathsf{Encap}, \mathcal{KEM}.\mathsf{Decap})$ (see Fig. 2).
The main ingredients required for our construction are as follows:

- An *entropic* post-challenge IND-CCA-BLT-secure PKE scheme $\mathsf{E\text{-}BLT} = (\mathsf{E\text{-}BLT.Setup}, \mathsf{E\text{-}BLT.Gen}, \mathsf{E\text{-}BLT.Enc}, \mathsf{E\text{-}BLT.Dec})$, that encrypts ν bit messages, and supports labels. Also, assume that $\mathsf{E\text{-}BLT}$ is entropic with respect to parameters (λ_{pre}, λ_{post}, t_{pre}, t_{post}) (refer to Definition 3).
- A (ϑ, ε) average-case (seedless) 2-source extractor $\mathsf{Ext2} : \{0,1\}^{\nu} \times \{0,1\}^{\nu} \to \{0,1\}^{u}$, with $\varepsilon = 2^{-u-\omega(\log \kappa)}$ (see Sect. 2.2 for its definition).
- A strong one-time signature (OTS) scheme $\mathcal{SS} = (\mathcal{SS}.\mathsf{Gen}, \mathcal{SS}.\mathsf{Sig}, \mathcal{SS}.\mathsf{Ver})$, with message space $\mathsf{poly}(\kappa)$ (see [6] for the definition of OTS).

Design Rationale: On a high level, to generate an encapsulated symmetric key, first we generate a key pair (vk, sk) of a one-time signature (OTS) scheme. We then use an *entropic* post-challenge IND-CCA-BLT secure PKE scheme ($\mathsf{E\text{-}BLT}$) to encrypt two random strings x_1 and x_2 independently with the verification key vk as the label/tag, and generate a signature on both the ciphertexts c_1 and c_2. The security of $\mathsf{E\text{-}BLT}$ guarantees that both the strings x_1 and x_2 still have enough average min-entropy after chosen-ciphertext leakage and tampering attacks (even in the post-challenge phase). In addition, the split-state model ensures that the two strings are independent. Hence, we can use an average-case seedless 2-source extractor to extract a random encapsulation key from both the strings. The trick of generating a key pair of an OTS and setting the verification key vk as a tag/label while encrypting, ensures that, a tag cannot be re-used by an adversary in a decryption or tampering query (In fact, to re-use that tag, the adversary essentially has to forge a signature under vk). The formal proof of our construction will follow this intuition, expect for one condition related to adaptivity of the adversary. The adversary may chose leakage and tampering functions from the two parts of the secret key after it saw the encapsulated key which was itself derived from the two parts, hence causing a circularity in the argument. This leap is handled in our proof using complexity leveraging. In particular, if the size of the extracted encapsulation key has u bits, then the adaptivity can only increase the advantage of the adversary by a factor at most 2^u. We set our parameters appropriately to handle this gap.

Theorem 3. *Let* $\mathsf{E\text{-}BLT}$ *be an entropic post-challenge* IND-CCA-BLT-*secure PKE scheme with parameters* (λ_{pre}, λ_{post}, t_{pre}, t_{post}) *and encrypting* ν *bit messages and supporting labels. Also, let* $\mathsf{Ext2}$ *be a* (ϑ, ε) *average-case (seedless) 2-source extractor with parameters mentioned above, and let* $\mathcal{SS}$ *be a strong one-time signature scheme supporting polynomial sized message space. Then the KEM scheme* $\mathcal{KEM}$ *is* IND-CCA *secure with respect to pre- and post-challenge leakage* λ'_{pre} *and* λ'_{post} *respectively, and pre- and post-challenge tampering* t'_{pre} *and* t'_{post} *respectively, in the bounded split-state leakage and tampering model, as long as the parameters satisfy the following constraints:*

$$\lambda'_{\mathsf{pre}} \leq \lambda_{\mathsf{pre}}, \;\; \lambda'_{\mathsf{post}} \leq \min(\lambda_{\mathsf{post}} - u, \, \nu - t'_{\mathsf{post}} \log p - \vartheta - 1), \;\; t'_{\mathsf{pre}} \leq t_{\mathsf{pre}} \;\text{ and }\; t'_{\mathsf{post}} \leq t_{\mathsf{post}}.$$

We refer the reader to the full version [6] for the detailed proof of the above theorem.

5 Post-challenge IND-CCA-BLT Secure PKE in Split-State Model

In this section, we present our construction of post-challenge IND-CCA-BLT secure PKE scheme in split-state model, starting from a post-challenge IND-CCA-BLT secure KEM scheme (as shown in Sect. 4.1) and a (one-time) symmetric-key encryption scheme. The security model of post-challenge IND-CCA-BLT secure PKE scheme in split state model is similar to the model of post-challenge IND-CCA-BLT secure KEM scheme in split state as described in Sect. 4, with the only difference that the encapsulation and the decapsulation algorithms are replaced by the encryption and decryption algorithms respectively. The secret key of the PKE is also split into two parts, as in the KEM scheme, and the adversary can query ask arbitrary pre- and post-challenge leakage and tampering queries, provided they act independently on the secret key

Define the key encapsulation scheme $\mathcal{KEM}$ as follows:

1. $\underline{\mathcal{KEM}.\mathsf{Setup}(1^\kappa)}$: On input 1^κ, run $\mathsf{E\text{-}BLT.SetUp}$ to get *params*. Set $\mathsf{par} := params$.
2. $\underline{\mathcal{KEM}.\mathsf{Gen}(\mathsf{par})}$: The key generation consists of two subroutines– $\mathcal{KEM}.\mathsf{Gen}_1$ and $\mathcal{KEM}.\mathsf{Gen}_2$, where $\mathcal{KEM}.\mathsf{Gen}_j$ on input par, samples $(pk_j, sk_j) \leftarrow \mathsf{E\text{-}BLT.Gen}(\mathsf{par})$, for $j = 1, 2$. It outputs the public key as $pk = (pk_1, pk_2)$, and the secret key is $sk = (sk_1, sk_2)$.
3. $\underline{\mathcal{KEM}.\mathsf{Encap}(pk)}$: On input the public key pk, do the following:
   - Run $(vk, ssk) \leftarrow \mathcal{SS}.\mathsf{Gen}(1^\kappa)$, where vk and ssk are the verification and signing keys of the strong OTS scheme respectively.
   - Choose $x_1, x_2 \xleftarrow{\$} \{0,1\}^\nu$ and compute $c_1 \leftarrow \mathsf{E\text{-}BLT.Enc}(pk_1, x_1, vk)$ and $c_2 \leftarrow \mathsf{E\text{-}BLT.Enc}(pk_2, x_2, vk)$, where vk is the label.
   - Compute $\sigma \leftarrow \mathcal{SS}.\mathsf{Sign}(ssk, (c_1, c_2))$ and $k = \mathsf{Ext2}(x_1, x_2)$.

   Output the ciphertext-key pair $(c = (vk, c_1, c_2, \sigma), k)$
4. $\underline{\mathcal{KEM}.\mathsf{Decap}(sk, c)}$: On input the secret key sk and the ciphertext c do:
   - Parse c as $c = (vk, c_1, c_2, \sigma)$ and $sk = (sk_1, sk_2)$.
   - Run $\mathcal{SS}.\mathsf{Ver}(vk, (c_1, c_2), \sigma)$. If the verification fails, the ciphertext is *invalid* and return $\perp$.
   - Run $x_j \leftarrow \mathsf{E\text{-}BLT.Dec}_j(sk_j, c_j)$ for $j = \{1, 2\}$.
   - Run $\mathcal{KEM}.\mathsf{Comb}(x_1, x_2)$: Compute $k = \mathsf{Ext2}(x_1, x_2)$.

Fig. 2. Post-challenge IND-CCA-BLT-secure KEM scheme $\mathcal{KEM}$.

parts and are bounded in length or number as before. Besides, he can ask arbitrary pre- and post-challenge decryption queries, with the obvious restriction that in the post-challenge phase the decryption queries are never asked on the challenge ciphertext. The challenge phase is replaced by the standard indistinguishability style definition for PKE scheme. The PKE scheme $\mathcal{BLT}$ consists of the following algorithms $\mathcal{BLT} = (\mathcal{BLT}.\mathsf{Setup}, \mathcal{BLT}.\mathsf{Gen}, \mathcal{BLT}.\mathsf{Enc}, \mathcal{BLT}.\mathsf{Dec})$. We refer the reader to [6] for the detailed model.

5.1 Construction of Post-challenge IND-CCA-BLT Secure PKE

We now show the construction of our post-challenge/after-the-fact IND-CCA-BLT secure PKE scheme $\mathcal{BLT} = (\mathcal{BLT}.\mathsf{Setup}, \mathcal{BLT}.\mathsf{Gen}, \mathcal{BLT}.\mathsf{Enc}, \mathcal{BLT}.\mathsf{Dec})$. The main ingredients of our construction are:

1. A 2-split-state IND-CCA-$\big(k, (\lambda'_{\mathsf{pre}}, \lambda'_{\mathsf{post}}), (t'_{\mathsf{pre}}, t'_{\mathsf{post}})\big)$-BLT secure KEM $\mathcal{KEM} = (\mathcal{KEM}.\mathsf{Setup}, \mathcal{KEM}.\mathsf{Gen}, \mathcal{KEM}.\mathsf{Encap}, \mathcal{KEM}.\mathsf{Decap})$ (please refer to [6] for the definition) with output space $\{0,1\}^* \times \{0,1\}^u$.
2. (One-time) symmetric encryption scheme $\varphi = (\mathcal{SKE}.\mathsf{KG}, \mathcal{SKE}.\mathsf{Enc}, \mathcal{SKE}.\mathsf{Dec})$ encrypting ω bit messages, with key space $\{0,1\}^u$. (please refer to [6] for its definition).

Construction: The construction of our 2-split-state PKE scheme $\mathcal{BLT}$ proceeds as follows:

1. $\mathcal{BLT}.\mathsf{Setup}(1^\kappa)$: Run $\mathsf{par} \leftarrow \mathcal{KEM}.\mathsf{Setup}(1^\kappa)$. Set $\mathsf{params} := \mathsf{par}$.
2. $\mathcal{BLT}.\mathsf{Gen}(\mathsf{params})$: Run $(pk, sk) \leftarrow \mathcal{KEM}.\mathsf{Gen}(\mathsf{par})$. Recall that public key $pk = (pk_1, pk_2)$ and $sk = (sk_1, sk_2)$. Set $pk' = pk$ and $sk' = sk$.
3. $\mathcal{BLT}.\mathsf{Enc}(pk', m)$: On input a message $m \in \{0,1\}^\omega$, run $(c_0, k) \leftarrow \mathcal{KEM}.\mathsf{Encap}(pk')$. Then it computes $c_1 \leftarrow \mathcal{SKE}.\mathsf{Enc}(k, m)$, and output the ciphertext $c = (c_0, c_1)$.
4. $\mathcal{BLT}.\mathsf{Dec}(sk', c)$: Parse $c = (c_0, c_1)$. Run $k \leftarrow \mathcal{KEM}.\mathsf{Decap}(sk', c_0)$, and outputs the message $m = \mathcal{SKE}.\mathsf{Dec}(k, c_1)$.

Theorem 4. *The encryption scheme $\mathcal{BLT}$ is post-challenge* IND-CCA-$(k, (\lambda''_{\mathsf{pre}}, \lambda''_{\mathsf{post}}), (t''_{\mathsf{pre}}, t''_{\mathsf{post}}))$-BLT *secure as long as the parameters satisfies:*

$$\lambda''_{\mathsf{pre}} \le \lambda'_{\mathsf{pre}}, \ \ \lambda''_{\mathsf{post}} \le \lambda'_{\mathsf{post}} \quad \text{and} \quad t''_{\mathsf{pre}} \le t'_{\mathsf{pre}}, \ \ t''_{\mathsf{post}} \le t'_{\mathsf{post}}$$

We refer the reader to the full version [6] for the detailed proof.

6 Conclusion

In this work, we study after-the-fact leakage and tampering in the context of public-key encryption schemes. To this end, we define an entropic post-challenge IND-CCA-BLT security and show how to construct full-fledged post-challenge IND-CCA-BLT secure PKE schemes under the split-state restriction. It is interesting to find other meaningful and realizable after-the-fact definitions of security for leakage and tampering. Besides, it will be interesting to define an appropriate framework for after-the-fact continuous leakage and tampering attacks, and port our construction in this setting.

Acknowledgments. We acknowledge the reviewers for their helpful comments. The authors are grateful to the project "Information Security Education and Awareness Program" of Ministry of Information Technology, Government of India for providing partial support.

References

1. Aggarwal, D., Dodis, Y., Kazana, T., Obremski, M.: Non-malleable reductions and applications. In: Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, pp. 459–468. ACM (2015)
2. Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous hardcore bits and cryptography against memory attacks. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 474–495. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00457-5_28
3. Bellare, M., Kohno, T.: A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 491–506. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-39200-9_31

4. Boneh, D., Halevi, S., Hamburg, M., Ostrovsky, R.: Circular-secure encryption from decision Diffie-Hellman. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 108–125. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85174-5_7
5. Chakraborty, S., Paul, G., Rangan, C.P.: Efficient compilers for after-the-fact leakage: from CPA to CCA-2 secure PKE to AKE. In: Pieprzyk, J., Suriadi, S. (eds.) ACISP 2017. LNCS, vol. 10342, pp. 343–362. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-60055-0_18
6. Chakraborty, S., Rangan, C.P.: Public key encryption resilient to post-challenge leakage and tampering attacks. Cryptology ePrint Archive, Report 2018/883 (2018). https://eprint.iacr.org/2018/883
7. Chor, B., Goldreich, O.: Unbiased bits from sources of weak randomness and probabilistic communication complexity. SIAM J. Comput. **17**(2), 230–261 (1988)
8. Damgård, I., Faust, S., Mukherjee, P., Venturi, D.: Bounded tamper resilience: how to go beyond the algebraic barrier. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013. LNCS, vol. 8270, pp. 140–160. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-42045-0_8
9. Dodis, Y., Haralambiev, K., López-Alt, A., Wichs, D.: Efficient public-key cryptography in the presence of key leakage. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 613–631. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17373-8_35
10. Dziembowski, S., Kazana, T., Obremski, M.: Non-malleable codes from two-source extractors. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 239–257. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_14
11. Dziembowski, S., Pietrzak, K., Wichs, D.: Non-malleable codes. In: ICS, pp. 434–452 (2010)
12. Faonio, A., Venturi, D.: Efficient public-key cryptography with bounded leakage and tamper resilience. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10031, pp. 877–907. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53887-6_32
13. Gennaro, R., Lysyanskaya, A., Malkin, T., Micali, S., Rabin, T.: Algorithmic tamper-proof (ATP) security: theoretical foundations for security against hardware tampering. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 258–277. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24638-1_15
14. Groth, J.: Simulation-sound NIZK proofs for a practical language and constant size group signatures. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 444–459. Springer, Heidelberg (2006). https://doi.org/10.1007/11935230_29
15. Halderman, J.A., et al.: Lest we remember: cold-boot attacks on encryption keys. Commun. ACM **52**(5), 91–98 (2009)
16. Halevi, S., Lin, H.: After-the-fact leakage in public-key encryption. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 107–124. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19571-6_8
17. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_25
18. Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-68697-5_9

19. Liu, F.-H., Lysyanskaya, A.: Tamper and leakage resilience in the split-state model. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 517–532. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_30
20. Naor, M., Segev, G.: Public-key cryptosystems resilient to key leakage. SIAM J. Comput. **41**(4), 772–814 (2012)
21. Santha, M., Vazirani, U.V.: Generating quasi-random sequences from semi-random sources. J. Comput. Syst. Sci. **33**(1), 75–87 (1986)
22. Vazirani, U.V.: Strong communication complexity or generating quasi-random sequences from two communicating semi-random sources. Combinatorica **7**(4), 375–392 (1987)
23. Zhang, Z., Chow, S.S., Cao, Z.: Post-challenge leakage in public-key encryption. Theor. Comput. Sci. **572**, 25–49 (2015)