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, 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

1 Introduction and Related Works

Traditionally, cryptographic schemes have been analyzed assuming that an adversary has only *black-box access* to the underlying functionality, and in no way is allowed to manipulate the internal state of the functionality. In particular, the adversary has only limited access to the cryptographic primitive/functionality via several different and well defined (restricted) interfaces/oracle queries. The security of the primitive holds only

in this idealistic setting as long as the adversarial access is limited to those that are defined in the security model. However, the real world physical implementation of a cryptosystem may leak much more unintended information, which can be exploited by an adversary to gain more insight into the system. These physical attacks can be broadly categorized into *passive* and *active* attacks. In the passive attack, 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 [18,21,22]. In the active attack scenario, 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 kind of 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. For instance, the attack in [4] recovers the RSA factors by introducing an arbitrary fault in one of two RSA signature computations. It is also shown in [25] that it is possible to retrieve the complete 128-bit secret key of AES with the help of only two faulty AES ciphertexts. See also [28] for the effect of optimal induction on the memory of a device.

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 allowable 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. [16], 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 [9], 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 [9], 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 [15] 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 [9, 15], 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. But, note that, this is not a limitation of the existing security notions or the constructions. Indeed, as shown in [19, 24], 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 [9, Section 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 [19] 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 [7, 30]. 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 [9, 15]. However, no solution to this issue was offered. Indeed it is mentioned in [15] 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 [9], 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 [19]. To take care of after-the-fact leakage, the authors in [19] 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 [1,12,14], 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 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. [9], augmented with post challenge leakage and tampering queries. The definition of restricted IND-CCA-BLT-security [9] 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 of [9], 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. [5] (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 [9]. 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 [9], in case of BHHO cryptosystem for each (pre-challenge) tampering query it is possible to simulate polynomially many decryption queries under it

[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.

by just leaking a single group element, thus reducing tampering to leakage. We use similar ideas as above 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. Also, note that, in our entropic notion of security we do not need the split-state leakage and tampering restriction. Instead, the adversary can query any arbitrary (but bounded) number of pre- and post-challenge leakage and tampering queries on the entire secret key. The reason this does not violate the impossibility result of [9] is due to the entropic restriction (please refer to Section 3 for the detailed model). We note that, the work of Faonio and Venturi [15] gives a comparatively efficient construction of IND-CCA-BLT secure PKE scheme compared to the work of Damgård et al [9]. 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. [9] 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 [9] 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 [19], although the PKE scheme of [19] 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

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

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 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 [15] (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 [23]. We observe that the framework of [23] 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 [23] 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 [23] 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 [23] is that, it considers the model of continual leakage and

tampering (in split-state), whereas our model considers bounded leakage and tampering (as in [9]) in split-state.

1.2 Organization

The rest of the paper is organized as follows. In section 2, we provide the necessary preliminaries required for our constructions. In section 3, we give our definition of entropic post-challenge IND-CCA-BLT secure PKE schemes and its restricted notion. In section 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 section 3.3. In section 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 section 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 $x \xleftarrow{\$} \mathcal{X}$ 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. We denote a distribution supported on $\{0,1\}^n$ with min-entropy k to be an (n, k)-source.

2.2 Basics of Information Theory.

Here we give some basic results related to information theory that will be needed throughout the paper.

Basic definitions related to Min-entropy

Definition 1. (Min-Entropy). *The min-entropy of a random variable X, denoted as $\mathrm{H}_\infty(\mathrm{X})$ is defined as $\mathrm{H}_\infty(\mathrm{X}) \stackrel{\text{def}}{=} -\log(\max_x \Pr[X = x])$. This is a standard notion of entropy used in cryptography, since it measures the worst-case predictability of X.*

Definition 2. (Average Conditional Min-Entropy). *The average-conditional min-entropy of a random variable X conditioned on a (possibly) correlated variable Z, denoted as $\widetilde{\mathrm{H}}_\infty(X|Z)$ is defined as*

$$\widetilde{\mathrm{H}}_\infty(X|Z) = \text{-} \log \left(\mathbb{E}_{z\leftarrow Z}\left[\max_x \Pr[X=x|Z=z]\right) = \text{-} \log \left(\mathbb{E}_{z\leftarrow Z}\left[2^{-\mathrm{H}_\infty(X|Z=z)}\right]\right).$$

This measures the worst-case predictability of X by an adversary that may observe a correlated variable Z.

We will make use of the following properties of average min-entropy.

Lemma 1. [11] *Let A, B, C be random variables. Then for any $\delta > 0$, the average conditional min-entropy $\mathrm{H}_\infty(A|(B=b))$ is at least $\widetilde{\mathrm{H}}_\infty(A|B) - \log(\frac{1}{\delta})$ with probability at least $1-\delta$ over the choice of b.*

Lemma 2. [11] *For any random variable X, Y and Z, if Y takes on values in $\{0,1\}^\ell$, then*

$$\widetilde{\mathrm{H}}_\infty(X|Y,Z) \geq \widetilde{\mathrm{H}}_\infty(X|Z) - \ell \quad \text{and} \quad \widetilde{\mathrm{H}}_\infty(X|Y) \geq \widetilde{\mathrm{H}}_\infty(X) - \ell.$$

Lemma 3. *Let X be a discrete random variable which is independent from the sequence of random variables $\{Y_1, Y_2, \cdots, Y_n\}$, i,e., $\widetilde{\mathrm{H}}_\infty(X\,|Y_1, Y_2, \cdots, Y_n) = \widetilde{\mathrm{H}}_\infty(X)$. Then for arbitrary functions $f_1, f_2, \ldots, f_n$ we also have*

$$\widetilde{\mathrm{H}}_\infty(X\,|\,Z_1, Z_2, \cdots, Z_n) = \widetilde{\mathrm{H}}_\infty(X), \quad \text{where} \quad Z_i = f_i(Y_i).$$

Lemma 4. [19] *Let X be a random variable with domain $\mathcal{X}$, and Z the random variable describing a uniformly sampled element from $\mathcal{X}$; and let Y be a random variable independent of X. For any $\varepsilon \in [0,1]$, if $\big|(X,Y) - (Z,Y)\big| \leq \varepsilon$, then*

$$\widetilde{\mathrm{H}}_\infty(X|Y) \geq -\log\left(\frac{1}{|\mathcal{X}|} + \varepsilon\right)$$

We also need the following fact about independent random variables.

Lemma 5. [13] *Let A, B be independent random variables. Consider a sequence $(V_1, \cdots, V_m)$ of random variables, where for some function ϕ, $V_i = \phi(V_1, \cdots, V_{i-1}, C_i)$, where C_i is either A or B. Then A and B are independent conditioned on $V_1, \cdots, V_m$.*

Definition 3. (Universal Hash Functions and the Leftover Hash Lemma). *A family of functions $\{H_x : \mathcal{Y} \to \{0,1\}^\ell\}_{x\in X}$ is universal if for all (n,k)-sources $\mathcal{Y}$, all $a \neq b \in \mathcal{Y}$, $\mathsf{Pr}_{x\in X}[H_x(a) = H_x(b)] = 2^{-\ell}$. Then, the leftover hash lemma states that, for any random variable W, and for a negligible ε we have :*

$$\big|(H_X(W), X) - (U_\ell, X)\big| \leq \frac{1}{2}\sqrt{2^{-\mathrm{H}_\infty(W)}2^\ell}, \quad \text{where } \ell \leq k - 2\log(\frac{1}{\varepsilon}) + 2.$$

2.3 Hash Proof Systems.

A hash-proof system (HPS) can be viewed as a key encapsulation mechanism (following [20,24]), in which the ciphertexts can be generated in two modes– *valid* and *invalid*. Given a public key and a valid ciphertext, the encapsulation key is well defined, and can be decapsulated using the secret key. Moreover, generating a valid ciphertext requires a witness corresponding to it. In contrast, invalid ciphertext essentially contains no information about the encapsulated key, i.e., given the public key and an invalid ciphertext, the distribution of the encapsulated key (output by the decapsulation algorithm) is statistically close to uniform. In addition, the HPS satisfies subset membership problem, i.e, the distribution of valid and invalid ciphertexts are computationally indistinguishable, even given the public key.

An ϵ-universal hash proof system HPS [20,24] $\mathcal{HPS} = (\mathsf{Gen}_{hps}, \mathsf{Pub}, \mathsf{Priv})$ has the following syntax: (i) Algorithm Gen_{hps} takes as input the security parameter κ, and outputs a set of public parameters $pub := (aux, \mathcal{C}, \mathcal{V}, \mathcal{K}, \mathcal{SK}, \mathcal{PK}, \Lambda_{(\cdot)} : \mathcal{C} \to \mathcal{K}, \mu : \mathcal{SK} \to \mathcal{PK})$, where aux might contain additional structural parameters, Λ_{sk} is a hash function and, for any $sk \in \mathcal{SK}$, the function $\mu(sk)$ defines the action of Λ_{sk} over the subset $\mathcal{V}$ of valid ciphertexts (i.e., Λ_{sk} is projective). Moreover the function Λ_{sk} is ϵ-almost universal:

Definition 4. *A projective hash function Λ_{sk} is ϵ-almost universal, if for all pk, $C \in \mathcal{C} \setminus \mathcal{V}$, and for all $K \in \mathcal{K}$, it holds that $\mathsf{Pr}(\Lambda_{sk}(C) = K | \mathbf{PK} = pk, C) \leq \epsilon$, where $\mathbf{SK}$ is uniform on $\mathcal{SK}$ conditioned on $\mathbf{PK} = \mu(\mathbf{SK})$*

(ii) Algorithm Pub takes as input a public key $pk = \mu(sk)$, a valid ciphertext $C \in \mathcal{V}$, and a witness w for $C \in \mathcal{V}$, and outputs the value $\Lambda_{sk}(C)$. (iii) Algorithm Priv take as input the secret key sk and a ciphertext $C \in \mathcal{C}$, and outputs the value $\Lambda_{sk}(C)$.

Definition 5. *A hash proof system $\mathcal{HPS}$ is ϵ-almost universal if the following holds:*

- *For all sufficiently large $\kappa \in \mathbb{N}$, and for all possible outcomes of Gen_{hps}, the underlying projective hash function is ϵ-almost universal.*
- *The underlying set membership problem is hard. Specifically, for any PPT adversary $\mathcal{A}_{\mathsf{SMP}}$ the following quantity is negligible:*

$$\mathsf{Adv}^{\mathrm{smp}}_{\mathcal{HPS},\mathcal{A}_{\mathsf{SMP}}}(\kappa) := \left|\mathsf{Pr}[\mathcal{A}(\mathcal{C}, \mathcal{V}, C_0) = 1 | C_0 \xleftarrow{\$} \mathcal{V}] - \mathsf{Pr}[\mathcal{A}(\mathcal{C}, \mathcal{V}, C_1) = 1 | C_1 \xleftarrow{\$} \mathcal{C} \setminus \mathcal{V}]\right.$$

The lemma below directly follows from the definition of hash-proof system and the notion of min-entropy.

Lemma 6. *Let $\Lambda_{(\cdot)}$ be an ϵ-almost universal. Then for all pk and $C \in \mathcal{C} \setminus \mathcal{V}$ it holds that $\widetilde{\mathrm{H}}_\infty(\Lambda_{\mathbf{SK}}(C) | \mathbf{PK} = pk, C) \geq -\log \epsilon$, where $\mathbf{SK}$ is uniform over $\mathcal{SK}$ conditioned on $\mathbf{PK} = \mu(\mathbf{SK})$.*

2.4 Two source Extractors.

In this section, we give an overview of two-source extractors [8,27,29] and their generalization, which will be required for our work.

Definition 6. (Seedless 2-source Extractor). *A function* $\mathsf{Ext2} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$ *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:* $\big|\ (\mathsf{Ext2}(X,Y) - U_m)\ \big| < \epsilon$*, where* U_m *refer to a uniform* m*-bit string.*

Definition 7. (Average-case Seedless 2-source Extractor). *A function* $\mathsf{Ext2} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$ *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* $\big|\ ((\mathsf{Ext2}(X,Y), Z) - (U_m, Z))\ \big| < \epsilon$*.*

It is also known that a worst-case 2-source extractor is also an average-case two-source extractor [19].

Lemma 7. [19] *For any* $\delta > 0$*, if* $\mathsf{Ext2} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$ *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.*

2.5 CCA-secure Symmetric-key Encryption

A symmetric-key encryption (SKE) scheme $\varphi = (\mathcal{SKE}.\mathsf{KG}, \mathcal{SKE}.\mathsf{Enc}, \mathcal{SKE}.\mathsf{Dec})$ consist of the key generation, encryption and decryption algorithms. We require perfect correctness to hold. We also want (standard) CCA-security to hold for the SKE scheme as defined in terms of a game between a challenger and an adversary.

1. *The challenger samples* $K \xleftarrow{\$} \mathcal{SKE}.\mathsf{KG}(\kappa)$.
2. *The adversary can make polynomially many decryption queries related to* K*. On input a ciphertext* C*, the challenger runs* $\mathcal{SKE}.\mathsf{Dec}(K, C)$*, and returns the answer to the adversary.*
3. *In the challenge phase, the adversary outputs two messages* m_0, m_1 *of equal length. The challenger chooses randomly a bit* $b \leftarrow \{0,1\}$*, computes* $C^* \leftarrow \mathcal{SKE}.\mathsf{Enc}(K, m_b)$*, and returns* C^* *to the adversary.*
4. *The adversary may continue to ask decryption queries* C *as long as* $C \neq C^*$*.*
5. *Finally, the adversary outputs a bit* b' *as a guess for the bit* b*.*

We define the advantage of the adversary $\mathcal{A}_{\mathsf{SKE}}$ in the above experiment as $\mathsf{Adv}^{\mathcal{SKE}}_{\varphi,\mathcal{A}_{\mathsf{SKE}}}(\kappa) = \mathsf{Pr}[b' = b] - \frac{1}{2}$.

2.6 Strong One-time signatures

A signature scheme $\mathcal{SS} = (\mathcal{SS}.\mathsf{Gen}, \mathcal{SS}.\mathsf{Sig}, \mathcal{SS}.\mathsf{Ver})$ is said to be *strongly unforgeable* against *one-time* chosen message attack, if no adversary with access to the signing oracle (with respect the verification key vk) only once, is able to come up with a message-signature pair (m^*, σ^*), such that $\mathcal{SS}.\mathsf{Ver}(m^*, \sigma^*) = 1$ and (m^*, σ^*) is not queried to the signing oracle, except with negligible probability. We refer the reader to [26] for construction of strong one-time signature scheme from one-way functions.

2.7 True Simulation Extractable Non-interactive Zero Knowledge Argument System

In this section, we recall the notion of (same-string) *true-simulation extractable non-interactive zero knowledge argument system* (tSE-NIZK) first introduced in [10] and also its extension to support labels/ tags. This notion is similar to the notion of simulation-sound extractable NIZKs [17] 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.

Let R be an efficiently computable binary relation. For pairs $(y, x) \in R$, we call y the statement and x the witness. Let $\mathbb{L} = \{y \mid \exists\, x \text{ s.t. } (y, x) \in R\}$ be the language consisting of statements in R. A NIZK argument system consists of three algorithms $(\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ such that: (1) Algorithm Gen takes as input 1^κ and generates a common reference string crs, a (simulation) trapdoor tk and an extraction key ek; (2) Algorithm P takes as input the statement-witness pair (y, x), crs and a label L, and outputs a proof π such that $(y, x) \in R$; (3) Algorithm V takes as input crs, a statement y, a label L and a purported proof π and output 1, if the proof is acceptable and 0 otherwise. We require $(\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ to satisfy the regular *completeness*, *adaptive soundness* and (composable) *zero-knowledge* properties. We denote the zero-knowledge simulator by Sim, which takes as input a label L, a statement y (and not the witness x) and the simulation trapdoor tk, and produces a simulated proof π.

Strong True-simulation Extractability: We start by defining the simulation oracle $\mathcal{SIM}_{\mathsf{tk}}(.,.)$. A query to the simulation oracle consists of a statement-witness pair (y, x), and a label L. The oracle checks if $(y, x) \in R$. If true, it outputs a simulated argument $\mathsf{Sim}(\mathsf{L}, \mathsf{tk}, y)$, otherwise it outputs $\perp$. There exists a PPT algorithm $\mathsf{Ext}(\mathsf{L}, y, \pi, \mathsf{ek})$ such that for all PPT adversaries $\mathcal{P}^*$, we have $\mathsf{Pr}[\mathcal{P}^* \text{ wins}| \leq \text{negl}(\kappa)$ in the following game.

1. *The challenger samples* $(\mathsf{crs}, \mathsf{tk}, \mathsf{ek}) \leftarrow \mathsf{Gen}(1^\kappa)$*, and gives* crs *to* $\mathcal{P}^*$.
2. $\mathcal{P}^{*\mathcal{SIM}_{\mathsf{tk}}(.,.)}$ *can adaptively access the simulation oracle* $\mathcal{SIM}_{\mathsf{tk}}(.,.)$.
3. *Finally, the adversary* $\mathcal{P}^*$ *outputs a tuple* $(y^*, \mathsf{L}^*, \pi^*)$.
4. *The challenger runs* $x^* \leftarrow \mathsf{Ext}(\mathsf{L}^*, y^*, \pi^*, \mathsf{ek})$
5. $\mathcal{P}^*$ *wins if* (a) $(y^*, \mathsf{L}^*, \pi^*) \neq (y, \mathsf{L}, \pi)$ *for all pairs* (y, L, π) *returned by the simulation oracle* $\mathcal{SIM}_{\mathsf{tk}}(.)$*;* (b) $\mathsf{V}(\mathsf{crs}, \mathsf{L}^*, y^*, \pi^*) = 1$ *and* (c) $(y^*, x^*) \notin R$.

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. [10] 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 section 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. [5] with appropriate parameters, satisfies our entropic restricted notion of security (see section 3.2). Finally, in section 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 the output length of f must be less than $|sk|$. Specifically, following the works of [2, 10], 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 8. (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 [9], 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.

[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.

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. **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

[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.

is used in the real game. Let the sets Q_{pre} and Q_{post} contain the tuples of the form $\{(\widetilde{m}_{i_1}, (c_{i_1}, L_{i_1})), \cdots, (\widetilde{m}_{i_{q(\kappa)}}, (c_{i_{q(\kappa)}}, L_{i_{q(\kappa)}}))\}_{i=1}^{t_{\mathsf{pre}}}$ 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. 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 9. **(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

[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.

respect to the working of (tampered) decryption oracle, as defined in the real game in def. 8. 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 Section 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 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 def 8. In particular, using the same notations from def. 8, 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) [5] is entropic restricted post-challenge IND-CCA-BLT secure. It was shown in [9] 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 8) 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 8 and showing a leakage to tamper reduction to take care of pre- and post-challenge tampering queries.

Lemma 8. *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 8 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*, 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 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 Appendix A.1.

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. Due to space constraints, we refer the reader to Appendix A.2 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.

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 $(pk, sk) \leftarrow \mathsf{ER\text{-}BLT.Gen}(params)$, and set $pk' = pk$, $sk' = sk$.
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 $r \xleftarrow{\$} \mathcal{R}$, 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 $\mathsf{E\text{-}BLT}$

Theorem 2. *Let* $\mathsf{ER\text{-}BLT}$ *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,* 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-$\big(k, (\lambda_{\mathsf{pre}}, \lambda_{\mathsf{post}}), (t_{\mathsf{pre}}, t_{\mathsf{post}})\big)$-BLT *secure PKE scheme.*

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 [9,15]. This can be achieved by (say) hard-coding the CRS in the encryption algorithm. We refer the reader to Appendix B for the detailed proof of this theorem.

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

In this section, we present a formal definition 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 [9,19,24], 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 split-state leakage and tampering restriction. Here, the the secret key of the cryptosystem is *split* into multiple disjoint parts (in our case only two), and the adversary can ask arbitrary leakage and tampering queries on each of these two parts *independently*, but not leak or tamper on the secret key parts jointly. However, the adversary is allowed to adaptively ask leakage/tampering functions depending on the answers of the previous queries. Before proceeding, we give the definition of 2-split-state tampering functions.

Definition 10. (Split-state (leakage/tampering) functions) *A function* $f:\{0,1\}^{2n}\rightarrow \{0,1\}^{2n}$ *is said to be a* 2-split-state *function if* $f(x)=(f_1(x_1), f_2(x_2))$*, for some functions* $f_1, f_2:\{0,1\}^n \rightarrow \{0,1\}^n$*, where* x_1*,* x_2 *are the first* n *and last* n *bits of* x *respectively. The functions* f_1 *and* f_2 *act independently on the bit strings* x_1 *and* x_2*. When this split-state functionality is used as leakage/tampering functions, we call them 2-split-state leakage/tampering functions.*

Definition 11. (Split state IND-CCA-BLT secure KEM). A 2-split state IND-CCA-BLT secure KEM scheme $\mathcal{KEM} = (\mathcal{KEM}.\mathsf{Setup}, \mathcal{KEM}.\mathsf{Gen}, \mathcal{KEM}.\mathsf{Encap}, \mathcal{KEM}.\mathsf{Decap})$ consists of the following algorithms:

- $\mathcal{KEM}.\mathsf{Setup}(1^\kappa)$*: The setup algorithm takes as input the security parameter, and output the public parameters* par*, which is taken as* (implicit) *input by all the algorithms.*
- $\mathcal{KEM}.\mathsf{Gen}(\mathsf{par})$*: The key generation algorithm comprises of two subroutines namely,* $\mathcal{KEM}.\mathsf{Gen}_1$ *and* $\mathcal{KEM}.\mathsf{Gen}_2$*. The subroutine* $\mathcal{KEM}.\mathsf{Gen}_i$ $(i \in \{1,2\})$ *generates the* i^{th} *public-secret key pair, i.e,* $(pk_i, sk_i) \leftarrow \mathcal{KEM}.\mathsf{Gen}_i(\mathsf{par}, r_i)$*; where* $r_i \in \{0,1\}^*$*. The public key consists of the pair* $pk = (pk_1, pk_2)$ *and the secret key consists of the pair* $sk = (sk_1, sk_2)$*.*
- $\mathcal{KEM}.\mathsf{Encap}(pk)$*: The* (randomized) *encapsulation algorithm takes as input a public key* $pk = (pk_1, pk_2)$*, and outputs a ciphertext-key pair* (c, k)*.*
- $\mathcal{KEM}.\mathsf{Decap}(sk = (sk_1, sk_2), c)$*: The decapsulation algorithm consists of two partial decapsulation subroutines* $\mathcal{KEM}.\mathsf{Decap}_1$*,* $\mathcal{KEM}.\mathsf{Decap}_2$*, and a combining subroutine* $\mathcal{KEM}.\mathsf{Comb}$*. The decapsulation subroutine* $\mathcal{KEM}.\mathsf{Decap}_i$ $(i \in \{1,2\})$ *takes as input the ciphertext* c*, the secret key split* sk_i *and outputs a partial decryption* t_i*, i.e.,* $t_i \leftarrow \mathcal{KEM}.\mathsf{Decap}_i(sk_i, c)$*. Finally,* $\mathcal{KEM}.\mathsf{Comb}$ *takes the ciphertext* c *and the pair* (t_1, t_2) *to recover the encapsulation key* k*, i.e.,* $k \leftarrow \mathcal{KEM}.\mathsf{Comb}(c, t = (t_1, t_2))$*.*

The *correctness* requirement of $\mathcal{KEM}$ states that $\forall \mathsf{par} \leftarrow \mathcal{KEM}.\mathsf{Setup}(1^\kappa)$, $(pk_i, sk_i) \leftarrow \mathcal{KEM}.\mathsf{Gen}_i(\mathsf{par})\,(i \in \{1,2\}), \forall (c,k) \leftarrow \mathcal{KEM}.\mathsf{Encap}(pk = (pk_1, pk_2))$, $\mathcal{KEM}.\mathsf{Decap}\big(sk = (sk_1, sk_2), c\big) = K$ holds with probability 1.

We now define the notion of CCA security of KEM schemes in the presence of after-the-fact split-state (bounded) memory leakage and tampering attacks.

Definition 12. (Post-Challenge IND-CCA-BLT security for KEM in split-state) *Let* $\kappa \in \mathbb{N}$ *be the security parameter. Let* $\lambda_{\mathsf{pre}}(\kappa)$ *and* $\lambda_{\mathsf{post}}(\kappa)$ *be the upper bound on the amount of memory leakage before and after the challenge phase respectively. Also, let* $t_{\mathsf{pre}}(\kappa)$ *and* $t_{\mathsf{post}}(\kappa)$ *be the maximum number of pre- and post-challenge tampering queries*

that can be asked by the adversary before and after the challenge phase respectively. A 2*-split-state KEM scheme* $\mathcal{KEM} = (\mathcal{KEM}.\mathsf{Setup}, \mathcal{KEM}.\mathsf{Gen}, \mathcal{KEM}.\mathsf{Encap}, \mathcal{KEM}.\mathsf{Decap})$ *is post-challenge* IND-CCA-$\big(k, (\lambda_{\mathsf{pre}}, \lambda_{\mathsf{post}}), (t_{\mathsf{pre}}, t_{\mathsf{post}})\big)$-BLT *secure if for all PPT adversaries* $\mathcal{B}_{\mathcal{KEM}}$*, the advantage* $\mathsf{Adv}^{\mathsf{AFL\text{-}IND\text{-}CCA\text{-}BLT}}_{\mathcal{B}_{\mathcal{KEM}},\mathcal{KEM}}(\kappa)$ *defined below is at most* $\frac{1}{2} + \mathsf{negl}(\kappa)$.

1. **Key Generation:** The challenger chooses $r_1, r_2 \xleftarrow{\$} \{0,1\}^*$, and compute $(pk_i, sk_i) \leftarrow \mathcal{KEM}.\mathsf{Gen}_i(\mathsf{par}, r_i)$ *(*$i \in \{1,2\}$*)* and sends $pk = (pk_1, pk_2)$ to the adversary, and keeps $sk = (sk_1, sk_2)$ to itself. Also, it initializes two lists $\mathcal{L}^1_{\mathsf{pre}} = \mathcal{L}^2_{\mathsf{pre}} = 0$, where $\mathcal{L}^i_{\mathsf{pre}}$ denotes the random variable quantifying the amount of leakage from the i^{th} split sk_i of the secret key sk ($i \in \{1,2\}$).
2. **Pre-Challenge Leakage:** The adversary makes an arbitrary number of leakage queries $(f^{\mathsf{pre}}_{1,i}, f^{\mathsf{pre}}_{2,i})$ adaptively, where $f^{\mathsf{pre}}_{1,i}$ and $f^{\mathsf{pre}}_{2,i}$ act independently on the secret key components sk_1 and sk_2 respectively. Upon receiving the i^{th} leakage query the challenger sends back $(f^{\mathsf{pre}}_{1,i}(sk_1), f^{\mathsf{pre}}_{2,i}(sk_2))$, provided $\mathcal{L}^1_{\mathsf{pre}} + |f^{\mathsf{pre}}_{1,i}(sk_1)| \leq \lambda_{\mathsf{pre}}(\kappa)$ and $\mathcal{L}^2_{\mathsf{post}} + |f^{\mathsf{pre}}_{2,i}(sk_2)| \leq \lambda_{\mathsf{pre}}(\kappa)$. It updates $\mathcal{L}^1_{\mathsf{pre}} = \mathcal{L}^1_{\mathsf{pre}} + |f^{\mathsf{pre}}_{1,i}(sk_1)|$, and $\mathcal{L}^2_{\mathsf{post}} = \mathcal{L}^2_{\mathsf{post}} + |f^{\mathsf{pre}}_{2,i}(sk_2)|$.
3. **Pre-Challenge Tampering:** The adversary is allowed to make at most t_{pre} number of pre-challenge tampering queries $(T^{\mathsf{pre}}_{1,i}, T^{\mathsf{pre}}_{2,i})$ ($i \in [t_{\mathsf{pre}}]$) adaptively. Each of the tampering functions $T^{\mathsf{pre}}_{1,i}$ and $T^{\mathsf{pre}}_{2,i}$ acts independently on the secret key splits sk_1 and sk_2 respectively. For each of the tampering query, the adversary $\mathcal{B}_{\mathcal{KEM}}$ gets access to the tampered decapsulation oracles $\mathcal{KEM}.\mathsf{Decap}(\widetilde{sk}_{1,\beta}, \cdot)$ and $\mathcal{KEM}.\mathsf{Decap}(\widetilde{sk}_{2,\beta}, \cdot)$, where $\widetilde{sk}_{\omega,\beta} = T^{\mathsf{pre}}_{\omega,\beta}(sk_\omega)$ (where $1 \leq \beta \leq i$, and $\omega \in \{1,2\}$). In other words, the decapsulation oracle may be queried with any of the tampered keys obtained till this point. We assume that, the total number of queries on the decapsulation oracles are polynomial. Note that, when $(T^{\mathsf{pre}}_{1,\beta}(sk_1), T^{\mathsf{pre}}_{2,\beta}(sk_2)) = (sk_1, sk_2)$, $\mathcal{B}_{\mathcal{KEM}}$ gets access to the (normal) decapsulation oracle in the pre-challenge phase.
4. **Challenge:** In this phase, the challenger computes $(c^*, k^*) \leftarrow \mathcal{KEM}.\mathsf{Encap}(pk)$. It then flips a uniform coin $b \xleftarrow{\$} \{0,1\}$. If $b = 0$, the challenger returns (c^*, k^*) to the adversary; otherwise it picks a key k^* uniformly at random, and sends sends (c^*, k^*) to the adversary.
5. **Post-Challenge Leakage:** The adversary makes an arbitrary number of leakage queries $(f^{\mathsf{post}}_{1,j}, f^{\mathsf{post}}_{2,j})$ adaptively, where $f^{\mathsf{post}}_{1,j}$ and $f^{\mathsf{post}}_{2,j}$ act independently on the secret key components sk_1 and sk_2 respectively. Upon receiving the j^{th} leakage query, the challenger sends back $(f^{\mathsf{post}}_{1,j}(sk_1), f^{\mathsf{post}}_{2,j}(sk_2))$, provided $\mathcal{L}^1_{\mathsf{post}} + |f^{\mathsf{post}}_{1,j}(sk_1)| \leq \lambda_{\mathsf{post}}(\kappa)$ and $\mathcal{L}^2_{\mathsf{post}} + |f^{\mathsf{post}}_{2,j}(sk_2)| \leq \lambda_{\mathsf{post}}(\kappa)$. It updates $\mathcal{L}^1_{\mathsf{post}} = \mathcal{L}^1_{\mathsf{post}} + |f^{\mathsf{post}}_{1,j}(sk_1)|$, and $\mathcal{L}^2_{\mathsf{post}} = \mathcal{L}^2_{\mathsf{post}} + |f^{\mathsf{post}}_{2,j}(sk_2)|$.
6. **Post-Challenge Tampering:** The adversary $\mathcal{B}_{\mathcal{KEM}}$ is allowed to make at most t_{post} number of post-challenge tampering queries $T_j = (T^{\mathsf{post}}_{1,j}, T^{\mathsf{post}}_{2,j})$ for $j \in [t_{\mathsf{post}}]$, where $\mathcal{B}_{\mathcal{KEM}}$ gets access to the tampered decapsulation oracles $\mathcal{KEM}.\mathsf{Decap}(\widetilde{sk}_{1,\gamma}, .)$ and $\mathcal{KEM}.\mathsf{Decap}(\widetilde{sk}_{2,\gamma}, .)$ respectively ($1 \leq \gamma \leq j$), as before. However, in the post-challenge phase, we impose an additional restriction that the adversary does not get access to the (tampered) decapsulation oracle(s) $\mathcal{KEM}.\mathsf{Decap}(\widetilde{sk}_{1,\gamma}, c^*)$ and $\mathcal{KEM}.\mathsf{Decap}(\widetilde{sk}_{2,\gamma}, c^*)$ with respect to the challenge ciphertext c^*.

7. **Guess:** Finally, the adversary $\mathcal{B}_{\mathcal{KEM}}$ outputs a bit b' for a guess of the bit b chosen the challenger. If $b' = b$, output 1, else output 0.

We define the advantage of the adversary $\mathcal{B}_{\mathcal{KEM}}$ in the above experiment as:

$$\mathsf{Adv}^{\mathsf{AFL\text{-}IND\text{-}CCA\text{-}BLT}}_{\mathcal{B}_{\mathcal{KEM}},\mathcal{KEM}}(\kappa) = \left|\mathsf{Pr}[b' = b] - \frac{1}{2}\right|.$$

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 Def 8).
- 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 Section 2.4 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 $\mathsf{poly}(\kappa)$.

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:*

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

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

1. $\mathcal{KEM}.\mathsf{Setup}(1^\kappa)$: On input 1^κ, run $\mathsf{E\text{-}BLT.SetUp}$ to get $params$. Set $\mathsf{par} := params$.
2. $\mathcal{KEM}.\mathsf{Gen}(\mathsf{par})$: 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. $\mathcal{KEM}.\mathsf{Encap}(pk)$: 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. $\mathcal{KEM}.\mathsf{Decap}(sk, c)$: 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 $\mathcal{KEM}$.

Proof. Let us denote the adversary for $\mathcal{KEM}$ by $\mathcal{B}_{\mathcal{KEM}}$ and let Simu be the entropic simulator for the underlying scheme $\mathsf{E\text{-}BLT}$. We need to show that no PPT adversary wins the post-challenge IND-CCA-BLT security game, except with negligible probability. In order to show this, we define a sequence of hybrids $\{\text{Hybrid}_i\}_{\{0 \leq i \leq 3\}}$, and show that the views of the adversary when interacting with these hybrids are computationally indistinguishable.

Before defining the hybrids, we assume that in the key generation phase, the challenger always first chooses a one-time signature key pair $(vk^*, ssk^*) \leftarrow \mathcal{SS}.\mathsf{Gen}(1^\kappa)$ and proceeds as in the hybrids described below. Then, in the challenge phase, instead of generating a OTS key pair $(vk, ssk) \leftarrow \mathcal{SS}.\mathsf{Gen}(1^\kappa)$, the challenger uses (vk^*, ssk^*) generated in the key generation phase.

Hybrid 0. This corresponds to the real game, expect with the (syntactical) modification as mentioned above. In particular, it chooses $x_1^*, x_2^* \xleftarrow{\$} \{0,1\}^\nu$. Then it computes $c_i^* \leftarrow \mathsf{E\text{-}BLT.Enc}(pk_i, x_i^*, vk^*)$ (for $i \in \{1,2\}$), $\sigma^* \leftarrow \mathcal{SS}.\mathsf{Sign}(ssk^*, (c_1^*, c_2^*))$ and $k^* = \mathsf{Ext2}(x_1^*, x_2^*)$. It then chooses a random bit $b \in \{0,1\}$. If $b = 0$, the encapsulated key is set to k^*; otherwise the challenger samples a random key $k^* \xleftarrow{\$} \{0,1\}^u$, and returns the challenge ciphertext-key pair $(c^* = (vk^*, c_1^*, c_2^*, \sigma^*), k^*)$ to $\mathcal{B}_{\mathcal{KEM}}$.

Hybrid 1. In this game, the challenger proceeds as in the $\mathbf{Hybrid}_0$, except that if the adversary $\mathcal{B}_{\mathcal{KEM}}$ submits a (normal) decapsulation query (vk, c_1, c_2, σ), with $vk = vk^*$, then it outputs $\perp$. As argued in Lemma 9, Hybrid 0 and Hybrid 1 are indistinguishable by the strong unforgeability property of the one-time signature scheme $\mathcal{SS}$.

Lemma 9. *For all PPT adversaries $\mathcal{B}_{\mathcal{KEM}}$ there exists a negligible function $\nu_{0,1}$ such that $\left|\mathsf{Adv}^{\mathsf{AFL\text{-}IND\text{-}CCA\text{-}BLT}}_{\mathcal{B}_{\mathcal{KEM}},\mathbf{Hybrid}_0}(\kappa) - \mathsf{Adv}^{\mathsf{AFL\text{-}IND\text{-}CCA\text{-}BLT}}_{\mathcal{B}_{\mathcal{KEM}},\mathbf{Hybrid}_1}(\kappa)\right| \leq \nu_{0,1}(\kappa)$.*

Proof. We observe that the views of the adversary $\mathcal{B}_{\mathcal{KEM}}$ in Hybrid_0 and Hybrid_1 are identical, except for an event F, which happens when $\mathcal{B}_{\mathcal{KEM}}$ submits a (normal) decapsulation query (vk, c_1, c_2, σ) with $vk = vk^*$. If $\mathcal{B}_{\mathcal{KEM}}$ submits such a valid decapsulation query in the pre-challenge phase, it trivially breaks the strong unforgeability of the OTS scheme. On the other hand, if $\mathcal{B}_{\mathcal{KEM}}$ comes up with such a query in the post-challenge phase, it must be the case that $c \neq c^*$, where c^* is the challenge ciphertext, thus contradicting the unforgeability of OTS again. The detailed proof of this lemma is shown below.

We will show that F happens with negligible probability, assuming the strong unforgeability property of the signature scheme $\mathcal{SS}$. In other words, if event F happens, we can construct a forger $\mathcal{F}$ against $\mathcal{SS}$. $\mathcal{F}$ simulates the environment for $\mathcal{B}_{\mathcal{KEM}}$ as follows.

$\mathcal{F}$ receives the verification key vk^* from the $\mathcal{SS}$ challenger (and hence $\mathcal{F}$ does not know ssk^*). In the pre-challenge phase, if $\mathcal{F}$ receives a decapsulation query of the form (vk^*, c_1, c_2, σ) such that $\mathcal{SS}.\mathsf{Ver}(vk^*, (c_1, c_2), \sigma) = 1$, or a pre-challenge tampering query of the form $(vk^*, c_{i,1}, c_{i,2}, \sigma_i)$ (with respect to the i^{th} tampered secret key) such that $\mathcal{SS}.\mathsf{Ver}(vk^*, (c_{i,1}, c_{i,2}), \sigma_i) = 1$, then $\mathcal{F}$ returns $\perp$ to the adversary $\mathcal{B}_{\mathcal{KEM}}$ and outputs $((c_1, c_2), \sigma)$ or $(c_{i,1}, c_{i,2}, \sigma_i)$ as the forgery. Otherwise, $\mathcal{F}$ proceeds as in Hybrid_0.
In the challenge phase, $\mathcal{F}$ chooses x_1^*, x_2^* uniformly at random and computes $c_1^* \leftarrow \mathsf{E\text{-}BLT.Enc}(pk_1, x_1^*, vk^*)$, $c_2^* \leftarrow \mathsf{E\text{-}BLT.Enc}(pk_2, x_2^*, vk^*)$. Then, it queries the signing oracle of $\mathcal{SS}$ for a signature σ^* on the message (c_1^*, c_2^*)[6]. It then chooses $b \leftarrow \{0,1\}$ randomly. If $b = 0$, it computes $k^* = \mathsf{Ext2}(x_1^*, x_2^*)$. Otherwise it chooses a string $k^* \xleftarrow{\$} \{0,1\}^u$ uniformly at random. Finally, $\mathcal{F}$ returns (c^*, k^*) to $\mathcal{B}_{\mathcal{KEM}}$.

In the post-challenge phase, if the adversary asks a decapsulation query of the form (vk^*, c_1, c_2, σ) such that $\mathcal{SS}.\mathsf{Ver}(vk^*, (c_1, c_2), \sigma) = 1$ and $(vk^*, c_1, c_2, \sigma) \neq (vk^*, c_1^*, c_2^*, \sigma^*)$, or a post-challenge tampering query of the form $(vk^*, c_{j,1}, c_{j,2}, \sigma_j)$ (with respect to the j^{th} tampered secret key) such that $\mathcal{SS}.\mathsf{Ver}(vk^*, (c_{j,1}, c_{j,2}), \sigma_j)$
$= 1$ and $(vk^*, c_{j,1}, c_{j,2}, \sigma_j) \neq (vk^*, c_1^*, c_2^*, \sigma^*)$, then $\mathcal{F}$ outputs $\perp$ to $\mathcal{B}_{\mathcal{KEM}}$ and returns $((c_1, c_2), \sigma)$ or $((c_{j,1}, c_{j,2}), \sigma_j)$ as the forgery, thus contradicting the strong-unforgeability property of the OTS $\mathcal{SS}$. □

[6] Note that the signing oracle is queried only once.

Hybrid 2. In this game, the challenger proceeds as in the $\mathbf{Hybrid}_1$, except that it uses the entropic simulator Simu to answer the first part of all the queries, i.e, to generate the ciphertext c_1^*, and answering leakage and tampering queries related to the secret key split sk_1. In more details the challenger does the following:

Key Generation: The challenger chooses x_1^* and x_2^* at random, then generates (pk_2, sk_2) using the key generation algorithm of $\mathsf{E\text{-}BLT}$, but it receives pk_1 by running $\mathsf{Simu}(x_1^*)$.

Pre-Challenge Leakage: When the adversary $\mathcal{B}_{\mathcal{KEM}}$ makes pre-challenge leakage queries $(f_{1,i}^{\mathsf{pre}}, f_{2,i}^{\mathsf{pre}})$, the challenger forwards the first part of the query $f_{1,i}^{\mathsf{pre}}$ to Simu and gets the answer from it. The challenge computes $f_{2,i}^{\mathsf{pre}}(sk_2)$ by itself, and returns both the answers to $\mathcal{B}_{\mathcal{KEM}}$.

Pre-Challenge Tampering: The pre-challenge tampering queries $(T_{1,i}^{\mathsf{pre}}, T_{2,i}^{\mathsf{pre}})$ made by $\mathcal{B}_{\mathcal{KEM}}$ are also answered similarly, i.e., $T_{1,i}^{\mathsf{pre}}$ is forwarded to Simu, and the challenge itself can compute $\widetilde{sk}_{2,i} = T_{2,i}(sk_2)$. When the adversary makes a decapsulation query $(vk, c_{1,i}, c_{2,i}, \sigma_i)$ with respect to the i^{th} tampered secret key $\widetilde{sk}_i = (\widetilde{sk}_{1,i}, \widetilde{sk}_{2,i})$, the challenger outputs $\perp$ if $vk = vk^*$. Otherwise, it first checks whether σ is a valid signature on $(c_{1,i}, c_{2,i})$ with respect to vk. If this is not the case, it outputs $\perp$. Otherwise, it asks Simu to decrypt $c_{1,i}$ under $\widetilde{sk}_{1,i}$, and receives the answer $\widetilde{x}_{1,i}$. The challenger then computes $\widetilde{x}_{2,i} \leftarrow \mathcal{KEM}.\mathsf{Decap}(\widetilde{sk}_{2,i}, c_{2,i})$ itself. Finally, it computes $\widetilde{k}_i \leftarrow \mathsf{Ext2}(\widetilde{x}_{1,i}, \widetilde{x}_{2,i})$, and returns $\widetilde{k}_i$ to $\mathcal{B}_{\mathcal{KEM}}$.

Challenge: In the challenge phase, the challenger asks Simu for the first ciphertext c_1^* under the label vk^* (chosen earlier), computes the second ciphertext $c_2^* \leftarrow \mathcal{KEM}.\mathsf{Encap}\big(pk_2, x_2^*, vk^*\big)$, and $\sigma^* \leftarrow \mathcal{SS}.\mathsf{Sign}\big(ssk^*, (c_1^*, c_2^*)\big)$. Next, the challenger makes a direct post-challenge leakage query to $\mathcal{KEM}.\mathsf{Decap}$, with the function $f_1(sk_1) = \mathsf{Ext2}\big(\mathcal{KEM}.\mathsf{Decap}(sk_1, c_1^*), x_2^*\big)$ (with output length u bits). Getting some answer k', the challenger just discards that answer, and instead computes $k^* = \mathsf{Ext2}(x_1^*, x_2^*)$ by itself. It then chooses a random bit b. If $b = 0$, the challenger sends $(c^* = (vk^*, c_1^*, c_2^*, \sigma^*), k^*)$ to the adversary $\mathcal{B}_{\mathcal{KEM}}$; otherwise it chooses k uniformly at random and sends $\big(c^* = (vk^*, c_1^*, c_2^*, \sigma^*), k\big)$ to $\mathcal{B}_{\mathcal{KEM}}$.

In the *post-challenge* phase, the leakage and tampering queries are handled identically as in the pre-challenge phase, with the challenger asking Simu for the first part of the answer (for queries related to sk_1) and answering the second part of the answer (queries related to sk_2) itself. Also, in the post-challenge phase, none of the decapsulation queries (with respect to the original secret key) should be equal to the challenge ciphertext c^*.

Note that the simulator Simu does not use the answers of the direct post-challenge leakage queries while computing the challenge ciphertext c^*. The reason that the challenger still makes them is to ensure that the entropic simulator see the same queries in this game as in the reductions shown below. One consequence of this query is that the entropic simulator has to answer more leakage queries than what is asked by the adversary $\mathcal{B}_{\mathcal{KEM}}$. More precisely, Simu has to answer extra u bits, hence we have $\lambda'_{\mathsf{post}} \leq \lambda_{\mathsf{post}} - u$.

We now prove that the event $b' = b$ holds in the above hybrids (**Hybrid** 1 and 2) with essentially the same probability as in the real game, by reducing to the indistinguishability property of the entropic simulator of $\mathsf{E\text{-}BLT}$.

Lemma 10. *For all PPT adversaries $\mathcal{B}_{\mathcal{KEM}}$ there exists a negligible function $\nu_{1,2}$ such that* $\left|\mathsf{Adv}^{\mathsf{AFL\text{-}IND\text{-}CCA\text{-}BLT}}_{\mathcal{B}_{\mathcal{KEM}},\mathbf{Hybrid}_1}(\kappa) - \mathsf{Adv}^{\mathsf{AFL\text{-}IND\text{-}CCA\text{-}BLT}}_{\mathcal{B}_{\mathcal{KEM}},\mathbf{Hybrid}_2}(\kappa)\right| \leq \nu_{1,2}(\kappa)$.

Proof. At a high level, the essential difference between these two hybrids is that: In **Hybrid$_2$**, the entropic simulator is used to simulate the first part of all the adversary's queries. However, unlike the challenger in Hybrid 2, the reduction here cannot discard the responses of the direct post-challenge leakage queries, since it does not know x_1^*, and hence cannot compute k^* by itself. To argue indistinguishability by reducing to the security of our entropic scheme $\mathsf{E\text{-}BLT}$, we must show that the responses of the post-challenge leakage queries are consistent in both Hybrid 2 and also in the reduction here. We do this by constructing two distinguishers $\mathcal{D}_1^{\mathsf{Ent}}$ and $\mathcal{D}_2^{\mathsf{Ent}}$ and prove that at least one of them has an advantage $\alpha/2$. The detailed proof of the lemma is shown below.

Assume that the event $b' = b$ holds in the Hybrid$_1$ with a probability that is larger than that in Hybrid$_2$ by a noticeable amount, say α. We describe an entropic adversary $\mathcal{A}^{\mathsf{Ent}}$ and two distinguishers $\mathcal{D}_1^{\mathsf{Ent}}$ and $\mathcal{D}_2^{\mathsf{Ent}}$ and prove that at least one of them has an advantage $\alpha/2$. $\mathcal{A}^{\mathsf{Ent}}$, $\mathcal{D}_1^{\mathsf{Ent}}$ and $\mathcal{D}_2^{\mathsf{Ent}}$ proceed as follows:

- The entropic adversary $\mathcal{A}^{\mathsf{Ent}}$ on input the public key pk_1, chooses (pk_2, sk_2) and x_2 in the same way as in Game 2, and sends (pk_1, pk_2) to $\mathcal{B}_{\mathcal{KEM}}$. It then proceeds similarly to the challenger in the real game, except in the challenge phase. More precisely, once $\mathcal{A}^{\mathsf{Ent}}$ receives c_1^* from the external oracle, it computes $c_2^* \leftarrow \mathsf{E\text{-}BLT.Enc}(pk_2, x_2^*, vk^*)$ and creates a signature $\sigma^* \leftarrow \mathcal{SS}.\mathsf{Sign}(ssk^*, (c_1^*, c_2^*))$. Then it makes a direct post-challenge leakage query asking for $k' = f_1(sk_1) = \mathsf{Ext2}(\mathcal{KEM}.\mathsf{Decap}(sk_1, c_1^*), x_2^*)$. As $\mathcal{A}^{\mathsf{Ent}}$ does not know x_1^*, it does not discard the answer; but instead uses k' (in case $b = 0$).
- The first distinguisher $\mathcal{D}_1^{\mathsf{Ent}}$ gets the view of $\mathcal{A}^{\mathsf{Ent}}$, which includes x_2^* and k' and also the string x_1^* (which is supposed to be encrypted in c_1^*). $\mathcal{D}_1^{\mathsf{Ent}}$ simply verifies if $k' \stackrel{?}{=} \mathsf{Ext2}(x_1^*, x_2^*)$. It outputs 1 if this verifies, else it outputs 0.
- The second distinguisher $\mathcal{D}_2^{\mathsf{Ent}}$ gets the view of $\mathcal{A}^{\mathsf{Ent}}$, which includes b and b', and outputs 1 if they are equal and 0 otherwise.

  If the oracle of $\mathcal{A}^{\mathsf{Ent}}$ is the real entropic post-challenge IND-CCA-BLT secure PKE $\mathsf{E\text{-}BLT}$, the value c_1^* is indeed a proper encryption x_1^* and the value k' is also equal to $\mathsf{Ext2}(x_1^*, x_2^*)$. Now, if the oracle of $\mathcal{A}^{\mathsf{Ent}}$ is the simulator Simu, there can be two possible cases: the event $k' \neq \mathsf{Ext2}(x_1^*, x_2^*)$ happens with probability at least $\alpha/2$ or with probability less than $\alpha/2$. In the first case, clearly the distinguisher $\mathcal{D}_1^{\mathsf{Ent}}$ has an advantage $\alpha/2$. In the second case, we will show that $\mathcal{D}_2^{\mathsf{Ent}}$ has advantage at least $\alpha/2$. Note that, in the second case, both the hybrids are identical except for an event that happens with probability less than $\alpha/2$. Since the probability of $b' = b$ in the real game is larger by α than this probability in Hybrid$_2$, then it is larger by more than α than this probability in the interaction with $\mathcal{A}^{\mathsf{Ent}}$. Hence the distinguisher $\mathcal{D}_2^{\mathsf{Ent}}$ has advantage more than $\alpha/2$. This proves that $k' = k$, except with negligible probability. □

Hybrid$_3$. This game is identical to Hybrid$_2$, except that, now both parts of the game are handled by the entropic simulator. The challenger runs $\mathsf{Simu}(x_1^*)$ and $\mathsf{Simu}(x_2^*)$ to generate the public keys pk_1 and pk_2, answer the leakage, tampering and decryption queries, and generate the challenge ciphertext. In the challenge phase, the challenger now makes direct post-challenge leakage queries to both copies of the simulator, asking the first for $k' = f_1(sk_1) = \mathsf{Ext2}(\mathcal{KEM}.\mathsf{Decap}(sk_1, c_1^*), x_2^*)$ and the second for $k'' = f_2(sk_2) = \mathsf{Ext2}(x_1^*,$
$\mathcal{KEM}.\mathsf{Decap}(sk_2, c_2^*))$. It then ignores both of the answers, and computes $k^* = \mathsf{Ext2}(x_1^*, x_2^*)$.

Finally, it chooses a random bit b and depending on b sends either $(c^* = (vk^*, c_1^*, c_2^*, \sigma^*), k^*)$ (in case $b = 0$) or $(c^* = (vk^*, c_1^*, c_2^*, \sigma^*), k)$ (in case $b = 1$) to the adversary $\mathcal{B}_{\mathcal{KEM}}$ as in Hybrid_2.

Lemma 11. *For all PPT adversaries $\mathcal{A}$ there exists a negligible function $\nu_{1,2}$ such that* $\left|\mathsf{Adv}^{\mathsf{AFL\text{-}IND\text{-}CCA\text{-}BLT}}_{\mathcal{B}_{\mathcal{KEM}},\mathbf{Hybrid}_2}(\kappa) - \mathsf{Adv}^{\mathsf{AFL\text{-}IND\text{-}CCA\text{-}BLT}}_{\mathcal{B}_{\mathcal{KEM}},\mathbf{Hybrid}_3}(\kappa)\right| \leq \nu_{2,3}(\kappa)$.

Proof. The proof of indistinguishability between Hybrid_2 and Hybrid_3 proceeds essentially in the same way as the proof of indistinguishability between Hybrid_1 and Hybrid_2. □

Advantage in Hybrid$_3$: We now estimate the advantage of $\mathcal{B}_{\mathcal{KEM}}$ in Hybrid_3.

Lemma 12. *The advantage of the adversary $\mathcal{B}_{\mathcal{KEM}}$ in* Hybrid_3 *is*

$$\mathsf{Adv}^{\mathsf{AFL\text{-}IND\text{-}CCA\text{-}BLT}}_{\mathcal{B}_{\mathcal{KEM}},\mathbf{Hybrid}_3}(\kappa) \leq \varepsilon \cdot 2^u.$$

For proving the above lemma, let us consider another related mental experiment:

$\overline{\mathbf{Hybrid_3}}$: This proceeds exactly as Hybrid_3, except that in the challenge phase, the challenger instead of sending $(c^* = (vk^*, c_1^*, c_2^*, \sigma^*), k^*)$ sends $c^* = (vk^*, c_1^*, c_2^*, \sigma^*)$, and defers sending k^* until after the post-challenge leakage and tampering phase.

Note that $\overline{\text{Hydrid}_3}$ is distinguishable from Hybrid_3. However, the following lemma shows that the difference only affects the advantage of the adversary by at most a multiplicative factor of 2^u.

Lemma 13. $\mathsf{Adv}^{\mathsf{AFL\text{-}IND\text{-}CCA\text{-}BLT}}_{\mathcal{B}_{\mathcal{KEM}},\overline{\mathbf{Hydrid_3}}}(\kappa) \leq \mathsf{Adv}^{\mathsf{AFL\text{-}IND\text{-}CCA\text{-}BLT}}_{\widetilde{\mathcal{B}}_{\mathcal{KEM}},\mathbf{Hybrid}_3}(\kappa) \cdot 2^u$

Proof. The proof of this lemma follows from standard complexity leveraging techniques. For every adversary $\mathcal{B}_{\mathcal{KEM}}$ in Hybrid_3, we present an adversary $\widetilde{\mathcal{B}}_{\mathcal{KEM}}$ for $\overline{\text{Hydrid}_3}$. The adversary $\widetilde{\mathcal{B}}_{\mathcal{KEM}}$ internally incorporates $\mathcal{B}_{\mathcal{KEM}}$ and forwards all the messages from $\mathcal{B}_{\mathcal{KEM}}$ to Hybrid_3 challenger, except in the challenge phase, where $\widetilde{\mathcal{B}}_{\mathcal{KEM}}$ randomly chooses a string $k^* \xleftarrow{\$} \{0,1\}^u$ and sends it to $\mathcal{B}_{\mathcal{KEM}}$. Later when $\widetilde{\mathcal{B}}_{\mathcal{KEM}}$ receives the real k^* from its challenger, it aborts if it guessed wrong, otherwise it proceeds just like $\mathcal{B}_{\mathcal{KEM}}$. Since the guess of $\widetilde{\mathcal{B}}_{\mathcal{KEM}}$ was correct with probability $1/2^u$, the statement in Lemma 13 follows. □

We now show that the advantage of adversary $\widetilde{\mathcal{B}}_{\mathcal{KEM}}$ in $\overline{\text{Hydrid}_3}$ is ε.

Lemma 14. *The advantage of an adversary $\widetilde{\mathcal{B}}_{\mathcal{KEM}}$ in* $\overline{\text{Hydrid}_3}$ *is:*

$$\mathsf{Adv}^{\mathsf{AFL\text{-}IND\text{-}CCA\text{-}BLT}}_{\widetilde{\mathcal{B}}_{\mathcal{KEM}},\overline{\mathbf{Hydrid_3}}}(\kappa) \leq \varepsilon$$

The proof of this follows from entropic security of $\mathsf{E\text{-}BLT}$, and the fact that the adversary does not learn k^*, except with negligible advantage. The split-state model then ensures that both x_1^* and x_2^* are independent, and hence the average-case (seedless) 2-source extractor $\mathsf{Ext2}$ guarantees that the output is ε-close to uniform.

Proof. Let Γ denote the transcript of messages that $\widetilde{\mathcal{B}}_{\mathcal{KEM}}$ received in $\overline{\text{Hydrid}_3}$ till the end of the post-challenge phase. We parse $\Gamma = (\Gamma_1, \Gamma_2, \Gamma_3)$, where Γ_1 contains the public key pk_1, the simulated encryption c_1^* of x_1^*, and all the leakage and tampering queries and responses on sk_1. Γ_2 contains the public key pk_2, the simulated encryption c_2^* of x_2^*, and all the leakage and tampering queries and responses on sk_2. Finally, Γ_3 contains the result of all the decryption oracle queries and responses (with respect to the original key). By the entropic property of $\mathsf{E\text{-}BLT}$, we have $\widetilde{\mathrm{H}}_\infty(x_1^*|(\Gamma_1, \Gamma_3)) \geq \nu - \lambda'_{\mathsf{post}} - t'_{\mathsf{post}} \log p - 1$ and $\widetilde{\mathrm{H}}_\infty(x_1^*|(\Gamma_2, \Gamma_3)) \geq \nu - \lambda'_{\mathsf{post}} - t'_{\mathsf{post}} \log p - 1$. Recall, that we set the parameter $\lambda'_{\mathsf{post}} \leq \nu - t'_{\mathsf{post}} \log p - \vartheta - 1$. So we get, $\widetilde{\mathrm{H}}_\infty(x_1^*|(\Gamma_1, \Gamma_3)) \geq \vartheta$. Moreover, as the random variables x_1 and Γ_2 are independent conditioned on Γ_1, and Γ_3, we also have $\widetilde{\mathrm{H}}_\infty(x_1^*|(\Gamma_2, \Gamma_3)) \geq \vartheta$. Similarly, we get $\widetilde{\mathrm{H}}_\infty(x_2^*|\Gamma) \geq \vartheta$

At this point, we can argue that the encapsulation key k^* is unpredictable, even given the public key, the challenge ciphertext and the answers to all the pre- and post-challenge leakage and tampering queries. Intuitively, after Hybrid 3 both c_1^* and c_2^* are simulated and hence by the entropic security of the underlying PKE scheme, both x_1^* and x_2^* (the underlying messages) retain high min-entropy, even conditioned on the pre- and post-challenge decryption, leakage and tampering queries, as shown above. Hence, at this stage, we can use the property of the 2-source extractor $\mathsf{Ext2}$ to argue that the encapsulated key k^* has high min-entropy even given all these information. More formally, we show that:

Lemma 15. $\widetilde{\mathrm{H}}_\infty(k^*|\,\Gamma) \geq \log(\frac{1}{\frac{1}{2^u}+\varepsilon})$, *where* $\Gamma = (\Gamma_1, \Gamma_2, \Gamma_3)$ *is the transcript of messages that* $\widetilde{\mathcal{B}}_{\mathcal{KEM}}$ *received in* $\overline{\text{Hydrid}_3}$ *till the end of the post-challenge phase, as defined above.*

Proof. The claim follows directly from the uniformity property of the average-case (seedless) 2-source extractor $\mathsf{Ext2}$. In particular, as shown above $\widetilde{\mathrm{H}}_\infty(x_1^*|\,\Gamma) \geq \vartheta$, and $\widetilde{\mathrm{H}}_\infty(x_2^*|\,\Gamma) \geq \vartheta$. Hence, by the (ϑ, ε) property of the extractor $\mathsf{Ext2}$, it follows that the value k^* has almost u bits of min-entropy. Specifically, by Lemma 4, we have that $\widetilde{\mathrm{H}}_\infty(k^*|\,\Gamma) \geq \log(\frac{1}{\frac{1}{2^u}+\varepsilon})$.

Parameter Setting. In the work of [6], it is shown how to construct a seedless $(2, t)$-non-malleable extractor for the following choice of the parameters:

Theorem 4 ([6]). *There exists a constant* $\gamma < 1/2$ *and a polynomial time computable function* $\mathsf{Ext} : \{0,1\}^t \times \{0,1\}^t \to \{0,1\}^{u'}$ *that is a* $(\vartheta, varepsilon)$*-2-source extractor at min-entropy* $\vartheta = \gamma t$ *with error* $\varepsilon = 2^{\Omega(u')}$, *and* $u' = \Omega(t)$.

According to Lemma 7, this is already an average-case extractor as needed. If we choose a length u' that is large enough than u, we can get $\varepsilon \leq 2^{-u-\omega(\log \kappa)}$. Then we can truncate the output to length u without increasing the statistical distance, thus getting the parameters that we need.

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 Section 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 Section 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 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 Appendix C 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 Definition 12) 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 $\{0,1\}^u$.

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 $pk = (pk_1, pk_2)$ 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 5. *The encryption scheme $\mathcal{BLT}$ is post-challenge* IND-CCA-$\big(k, (\lambda''_{\mathsf{pre}}, \lambda''_{\mathsf{post}}), (t''_{\mathsf{pre}}, t''_{\mathsf{post}})\big)$-BLT *secure as long as the parameters satisfies:*

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

Proof Sketch. We now sketch the main ideas for proving the above theorem. The proof follows mainly from the security of our KEM scheme $\mathcal{KEM}$, and the CCA-security of the SKE scheme φ. In particular, the leakage and tampering queries of the SKE adversary is handled by the leakage and tamper resilience of our KEM scheme. However, the adversary may also ask additional decryption queries with respect to the tampered keys, even on the challenge ciphertext. For this, we resort to the entropic property of $\mathcal{KEM}$. The IND-CCA-BLT security of $\mathcal{KEM}$ guarantees that the challenge KEM key k^* is close to uniform, even given the tampered keys. In the intermediate hybrid, we sample k^* uniformly at random, and show that if the adversary can distinguish between the real game and this hybrid, it can break the IND-CCA-BLT security of $\mathcal{KEM}$. Now the KEM key is independent of the tampered keys. When the SKE adversary queries the decryption oracle with respect to the tampered keys, the reduction asks for the corresponding tampered keys from the KEM challenger, and then it can answer the decryption queries by itself. Since the reduction can simulate the responses of the decryption oracle(s) by itself, the min-entropy of k^* is preserved. At this point, we use the (standard) IND-CCA security of our SKE scheme φ to argue indistinguishability of the challenge ciphertext. We refer the reader to Appendix D 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.

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: Theory of Cryptography Conference. pp. 474–495. Springer (2009)
3. Bellare, M., Kohno, T.: A theoretical treatment of related-key attacks: Rka-prps, rka-prfs, and applications. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 491–506. Springer (2003)
4. Boneh, D., DeMillo, R.A., Lipton, R.J.: On the importance of eliminating errors in cryptographic computations. Journal of cryptology 14(2), 101–119 (2001)
5. Boneh, D., Halevi, S., Hamburg, M., Ostrovsky, R.: Circular-secure encryption from decision diffie-hellman. In: Annual International Cryptology Conference. pp. 108–125. Springer (2008)
6. Bourgain, J.: More on the sum-product phenomenon in prime fields and its applications. International Journal of Number Theory 1(01), 1–32 (2005)

7. Chakraborty, S., Paul, G., Rangan, C.P.: Efficient compilers for after-the-fact leakage: from cpa to cca-2 secure pke to ake. In: Australasian Conference on Information Security and Privacy. pp. 343–362. Springer (2017)
8. Chor, B., Goldreich, O.: Unbiased bits from sources of weak randomness and probabilistic communication complexity. SIAM Journal on Computing 17(2), 230–261 (1988)
9. Damgård, I., Faust, S., Mukherjee, P., Venturi, D.: Bounded tamper resilience: How to go beyond the algebraic barrier. In: Advances in Cryptology–ASIACRYPT 2013: 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013. pp. 140–160. Springer (2013)
10. Dodis, Y., Haralambiev, K., López-Alt, A., Wichs, D.: Efficient public-key cryptography in the presence of key leakage. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 613–631. Springer (2010)
11. Dodis, Y., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In: International conference on the theory and applications of cryptographic techniques. pp. 523–540. Springer (2004)
12. Dziembowski, S., Kazana, T., Obremski, M.: Non-malleable codes from two-source extractors. In: Advances in Cryptology–CRYPTO 2013, pp. 239–257. Springer (2013)
13. Dziembowski, S., Pietrzak, K.: Intrusion-resilient secret sharing. In: Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on. pp. 227–237. IEEE (2007)
14. Dziembowski, S., Pietrzak, K., Wichs, D.: Non-malleable codes. In: ICS. pp. 434–452 (2010)
15. Faonio, A., Venturi, D.: Efficient public-key cryptography with bounded leakage and tamper resilience. In: Advances in Cryptology–ASIACRYPT 2016: 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I 22. pp. 877–907. Springer (2016)
16. Gennaro, R., Lysyanskaya, A., Malkin, T., Micali, S., Rabin, T.: Algorithmic tamper-proof (atp) security: Theoretical foundations for security against hardware tampering. In: Theory of Cryptography Conference. pp. 258–277. Springer (2004)
17. Groth, J.: Simulation-sound nizk proofs for a practical language and constant size group signatures. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 444–459. Springer (2006)
18. Halderman, J.A., Schoen, S.D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J.A., Feldman, A.J., Appelbaum, J., Felten, E.W.: Lest we remember: cold-boot attacks on encryption keys. Communications of the ACM 52(5), 91–98 (2009)
19. Halevi, S., Lin, H.: After-the-fact leakage in public-key encryption. In: Theory of Cryptography Conference. pp. 107–124. Springer (2011)
20. Kiltz, E., Pietrzak, K., Stam, M., Yung, M.: A new randomness extraction paradigm for hybrid encryption. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 590–609. Springer (2009)
21. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Advances in Cryptology-CRYPTO99. pp. 388–397. Springer (1999)
22. Kocher, P.C.: Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In: Advances in CryptologyCRYPTO96. pp. 104–113. Springer (1996)
23. Liu, F.H., Lysyanskaya, A.: Tamper and leakage resilience in the split-state model. In: Advances in Cryptology–CRYPTO 2012, pp. 517–532. Springer (2012)
24. Naor, M., Segev, G.: Public-key cryptosystems resilient to key leakage. SIAM Journal on Computing 41(4), 772–814 (2012)
25. Piret, G., Quisquater, J.J.: A differential fault attack technique against spn structures, with application to the aes and khazad. In: International Workshop on Cryptographic Hardware and Embedded Systems. pp. 77–88. Springer (2003)
26. Rompel, J.: One-way functions are necessary and sufficient for secure signatures. In: Proceedings of the twenty-second annual ACM symposium on Theory of computing. pp. 387–394. ACM (1990)

27. Santha, M., Vazirani, U.V.: Generating quasi-random sequences from semi-random sources. Journal of Computer and System Sciences 33(1), 75–87 (1986)
28. Skorobogatov, S.P., Anderson, R.J.: Optical fault induction attacks. In: International workshop on cryptographic hardware and embedded systems. pp. 2–12. Springer (2002)
29. Vazirani, U.V.: Strong communication complexity or generating quasi-random sequences from two communicating semi-random sources. Combinatorica 7(4), 375–392 (1987)
30. Zhang, Z., Chow, S.S., Cao, Z.: Post-challenge leakage in public-key encryption. Theoretical Computer Science 572, 25–49 (2015)

SUPPLEMENTARY MATERIALS

A Detailed proof of Theorem 1

A.1 Proof of Lemma 8

Lemma 7: *The BHHO cryptosystem described in Section 3.2 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 8 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 BHHO encryption scheme can be viewed as a *hash proof system* with the space of all ciphertext and valid ciphertexts defined as follows:

$$\mathcal{C} = \{(g_1^{r_1}, \cdots, g_\ell^{r_\ell}) : r_1, \cdots, r_\ell \in \mathbb{Z}_p\}$$
$$\mathcal{V} = \{(g_1^{r}, \cdots, g_\ell^{r}) : r \in \mathbb{Z}_p\}$$

The public and the private evaluation algorithms are as follows:

$$\mathsf{Pub}(pk, c, w) = \mathsf{Pub}(h, g^{\boldsymbol{\alpha}}, r) = h^r = \prod_{i=1}^{\ell} \left(g_i^{x_i}\right)^r$$

$$\mathsf{Priv}(sk, c = (\underbrace{c_1, \cdots, c_\ell}_{\mathbf{c}'}, d)) = \mathsf{Priv}(sk = \mathbf{x}, \mathbf{c}' = (g_1^r, \cdots, g_\ell^r)) = \prod_{i=1}^{\ell} (g_i^r)^{x_i} = h^r$$

Also note, that hardness of the subset membership problem between the valid and invalid ciphertexts follows easily from the DDH assumption.

Real game: In the real game, the challenger generates the public parameters *params* and the public-secret key pair $(pk = (h, g^{\boldsymbol{\alpha}}), sk = \mathbf{x})$ using the algorithms ER-BLT.SetUp and ER-BLT.Gen respectively. It then gives *params* and pk to the adversary. The answers to pre- and post-challenge leakage queries are answered by the challenger using the secret key sk. Note that the adversary does not ask any tampering query in this game. In the challenge phase, the challenger constructs the challenge ciphertext c^* as in the BHHO construction. In other words, it chooses a valid ciphertext $c^* \in \mathcal{V}$ by sampling $r \xleftarrow{\$} \mathbb{Z}_p$, and computing $\mathbf{c}' = (g_1^r, \cdots, g_\ell^r)$, and $c^* = (\mathbf{c}', \mathsf{Pub}(h, g^{\boldsymbol{\alpha}}, r) \cdot m) = (\mathbf{c}', h^r \cdot m)$, where m is the message chosen by the challenger at the onset of the entropic game. Finally, its outputs c^* to the adversary.

Simulated game: In our case, the simulator proceeds identically as the challenger in the real game, except in the generation of the challenge ciphertext, it chooses an invalid ciphertext $c^* \in \mathcal{C} \backslash \mathcal{V}$, i.e., it samples $r_1, \cdots, r_\ell \xleftarrow{\$} \mathbb{Z}_p$, and compute $\mathbf{c}' = (g_1^{r_1}, \cdots, g_\ell^{r_\ell})$, and $c^* = (\mathbf{c}', \mathsf{Priv}(\mathbf{x}, \mathbf{c}') \cdot m)$, where $\mathsf{Priv}(\mathbf{x}, \mathbf{c}') = \prod_{i=1}^{\ell} (g_i^{r_i})^{x_i}$. Finally it outputs c^* to the adversary.

It follows directly from the hardness of the subset membership problem of the above hash proof system that the real game is indistinguishable from the simulated game. It only remains to show that the min-entropy of the message is preserved, even given pre- and post-challenge leakage.

Note that the min-entropy of the secret key $sk(=\mathbf{x})$ is $\ell \log p$ (since each $x_i \in \mathbb{Z}_p$), and hence the average min-entropy of sk given the public key pk and pre-challenge leakage is:

$$\widetilde{\mathrm{H}}_\infty(\mathbf{x}|pk, \mathcal{L}_{\mathsf{pre}}) \geq \ell \log p - \log p - \lambda'_{\mathsf{pre}} \geq \log p + \omega(\log \kappa) + \lambda'_{\mathsf{post}}$$

where the last equation follows from our choice of $\ell \geq 2 + \frac{\lambda'_{\mathsf{pre}} + \lambda'_{\mathsf{post}} + \omega(\log \kappa)}{\log p}$.
Therefore, the leftover hash lemma guarantees that with overwhelming probability over the choice of $\mathbf{c}' = (c_1, \cdots, c_\ell) \in \mathcal{C} \backslash \mathcal{V}$, where $c_i = g_i^{r_i}$, it holds that $\Lambda_{sk}(\mathbf{c}') = \prod_{i=1}^{\ell} c_i^{x_i}$ is ε-close to the uniform distribution over $\mathbb{G}$, even given $h = \prod_{i=1}^{\ell} g_i^{x_i}$, and any leakage of λ'_{pre} bits. We refer the reader to [5] for more detailed discussion of the application of the leftover hash lemma in this context. This proves the *ε-smoothness* property of the hash proof system. Hence the message m has almost $\log p$-bits of entropy. In particular it has at least $\log(\frac{1}{\frac{1}{|\mathbb{Z}_p|} + \varepsilon})$ bits of min-entropy. Note that, for application of the leftover hash lemma, it is sufficient if the average conditional min-entropy of the secret key sk, given h and the pre-challenge leakage is at least $\log p + \omega(\log \kappa)$. Next, by relying on the fact that the post challenge leakage is bounded by λ'_{post} bits, we get:

$$\widetilde{\mathrm{H}}_\infty(\mathbf{x}|pk, \mathcal{L}_{\mathsf{pre}}, \mathcal{L}_{\mathsf{post}}) \geq \log p + \omega(\log \kappa) + \lambda'_{\mathsf{post}} - \lambda'_{\mathsf{post}} \geq \log p + \omega(\log \kappa)$$

Hence, even after the pre- and post-challenge leakage, the secret key retains enough min-entropy. So, the leftover hash lemma can again be applied in this context and hence the value $\Lambda_{sk}(\mathbf{c}')$ is ε-close to uniform distribution over $\mathbb{G}$, even given h, and pre- and post-challenge leakages of λ'_{pre} and λ'_{post} bits respectively. Hence the message m also retains a high min-entropy. This completes the proof of Lemma 8. □

A.2 Proof of Theorem 1 continued:

Proof. Let us assume 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 probability greater than $1/p(\kappa)$, for some polynomial $p(\kappa)$. We can 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. This leads to a contradiction of Lemma 8, which we just proved above. The reduction follows the steps described below:

1. The adversary $\mathcal{A}'$ receives $params$ and $(pk = (h, g^{\boldsymbol{\alpha}})$ from its challenger, and forwards $(params, pk)$ to $\mathcal{A}$. The challenger also chooses a message $m \leftarrow U_k$, and keeps it with itself.
2. The leakage queries asked by $\mathcal{A}$, both in the pre- and post-challenge phase, are answered by $\mathcal{A}'$ by forwarding the leakage queries to its leakage oracle, and returning the answer to $\mathcal{A}$.
3. The way that $\mathcal{A}'$ answers to the pre- and post-challenge tampering queries of $\mathcal{A}$ are as follows:

   On input a tampering query $T_i \in \mathcal{T}_{sk}$, $\mathcal{A}'$ submits a leakage query in order to retrieve the value $\widetilde{h}_i = \prod_{i=1}^{\ell} g_j^{-\widetilde{x}_{j,i}}$, where $\widetilde{\mathbf{x}}_i = T_i(\mathbf{x}) = (\widetilde{x}_{1,i}, \cdots, \widetilde{x}_{\ell,i})$ is the i^{th} tampered

secret key. When $\mathcal{A}$ makes a decryption query of the form (m, r)[7], $\mathcal{A}'$ returns $\widetilde{m} = (h^r \cdot m)\widetilde{h}_i^r$. Note that this is a perfect simulation, since $\mathcal{A}'$ produces a distribution which is identical to the one which $\mathcal{A}$ expects. This is shown below.

$$\widetilde{m} = (h^r \cdot m)\widetilde{h}_i^r = d\left(\cdot \prod_{j=1}^{\ell} g_j^{-\widetilde{x}_{j,i}} \right)^r = d / \prod_{j=1}^{\ell} (g_j^r)^{\widetilde{x}_{j,i}} \quad \text{where} \quad d = h^r \cdot m$$

This perfectly simulates the answers of the (modified) decryption oracle $\mathsf{Dec}^*(\widetilde{sk}_i, \cdot, \cdot)$, that $\mathcal{A}$ is supposed to receive.

4. In the challenge phase, $\mathcal{A}'$ asks its challenger for the challenge ciphertext c^*. It then returns c^* to $\mathcal{A}$.
5. Finally, $\mathcal{A}'$ outputs whatever $\mathcal{A}$ outputs.

Note that in the above reduction, all the pre- and post-challenge tampering queries of $\mathcal{A}$ are handled by $\mathcal{A}'$ by leaking one group element for each tampering query. Hence we have,

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

where t_{pre} and t_{post} are the number of pre- and post-challenge tampering queries made by $\mathcal{A}$. This concludes the proof of Theorem 1. □

B Proof of Theorem 2

Theorem 2. *Let* ER-BLT *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,* 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* E-BLT *is an entropic post-challenge* IND-CCA-$\big(k, (\lambda_{\mathsf{pre}}, \lambda_{\mathsf{post}}), (t_{\mathsf{pre}}, t_{\mathsf{post}})\big)$-BLT *secure PKE scheme.*

Proof. Let S_{ER} and S_{E} denote the entropic simulators of the entropic restricted and the entropic post-challenge IND-CCA-BLT PKE respectively. According to Definition 9 we need to show that on one hand S_{E} answers queries from the adversary in an indistinguishable way from real game and on the other hand leaves the challenge message with high min-entropy.

Setup and Key Generation: The simulator S_{E} on input a uniformly chosen message m^* invokes the simulator $\mathsf{S}_{\mathsf{ER}}(m^*)$, and receives the public parameters $params$ and the public key pk from it. S_{E} then runs $(\mathsf{crs}, \mathsf{tk}, \mathsf{ek}) \leftarrow \mathsf{Gen}(1^\kappa)$, sets $params' = (params, \mathsf{crs})$. Further, it also sets $pk' = pk$. It then sends $(params', pk')$ to the adversary, and keeps the trapdoor informations $(\mathsf{tk}, \mathsf{ek})$ for the NIZK with itself.

Phase 1: In this phase the adversary $\mathcal{A}$ makes the following queries to S_{E}:

– *Leakage queries*: Here, the adversary asks for a leakage from the secret key by specifying a polynomial time computable function $f^{\mathsf{pre}}(.)$. If the output length of f^{pre} is no more than λ_{pre} bits, the challenger forwards this to S_{ER}, returns the answer ans to $\mathcal{A}$. It also sets $L_{\mathsf{pre}} := L_{\mathsf{pre}} + |ans|$. Otherwise, it ignores the query.

[7] Note in the entropic restricted game the adversary has access only to the modified tampering oracle, defined in section 3.1.

– *Tampering queries*: The adversary may ask pre-challenge tampering queries $T_i \in \mathcal{T}_{sk}$, where $i \in [t_{\mathsf{pre}}]$. The challenger forwards T_i to S_{ER}. When $\mathcal{A}$ asks a decryption query $c'_i = (c_i, \pi_i, \mathsf{L}_i)$ with respect to the i^{th} tampered secret key[8], the challenger S_{E} checks if the proof π_i is accepting with respect to the label L_i. If it accepts, it runs the extractor of the tSE-NIZK to output (m_i, r_i), i.e., $(m_i, r_i) \leftarrow \mathsf{Ext}(\mathsf{L}_i, (pk, c_i), \pi_i, \mathsf{ek})$, and forwards the pair (m_i, r_i) to its challenger[9], which returns $\widetilde{m}_i$. S_{E} then returns $\widetilde{m}_i$ to $\mathcal{A}$.
– *Decryption queries*: The adversary can make polynomially many decryption queries with respect to the original secret key. On a decryption query $c' = (c, \pi, \mathsf{L})$, S_{E} checks if the proof π is accepting with respect to L. If it accepts, it runs $(m, r) \leftarrow \mathsf{Ext}(\mathsf{L}, (pk, c), \pi, \mathsf{ek})$, and returns the message m to $\mathcal{A}$.

Challenge: In this phase, the adversary submits a bit-string L^* as a label. S_{E} asks for a challenge ciphertext from its challenger, and receives c^* in response. It then generates a simulated proof by using the zero-knowledge simulator, i.e., $\pi^* \leftarrow \mathsf{Sim}(\mathsf{L}^*, (pk, c^*), \mathsf{tk})$. It then returns the ciphertext $c'^* = (c^*, \pi^*)$ to the adversary $\mathcal{A}$.

Phase 2: In this phase, the adversary $\mathcal{A}$ also makes leakage, tampering and decryption queries, similar to phase 1.

– *Leakage queries*: Here, the adversary asks for a leakage from the secret key by specifying a polynomial time computable function $f^{\mathsf{post}}(.)$. If the output length of f^{post} is no more than λ_{post} bits, the challenger forwards this to S_{ER}, returns the answer ans to $\mathcal{A}$. It also sets $L_{\mathsf{post}} := L_{\mathsf{post}} + |ans|$. Otherwise, it ignores the query.
– *Tampering queries*: The adversary may also ask post-challenge tampering queries $T_j \in \mathcal{T}_{sk}$, where $j \in [t_{\mathsf{post}}]$. The way the tampering queries of $\mathcal{A}$ are answered is similar to phase 1.
– *Decryption queries*: The decryption queries are also answered by S_{E} similar to phase 1, if $(c, \mathsf{L}) \neq (c'^*, \mathsf{L}^*)$, else it outputs $\perp$.

Next, we show that the view of the adversary $\mathsf{View}^{\mathsf{sm}}_{\mathsf{S}_{\mathsf{E}},\mathcal{A}}(\kappa)$ in the simulated game above is indistinguishable from the real view, even given the message m^*. We define a sequence of hybrids $\{\mathsf{Hyb}_i\}_{i\in[3]}$, and show that the adversary cannot distinguish between them, except with negligible probability. The last hybrid Hyb_3 is identical to the simulated game, and hence the view of the adversary produced by Hyb_3 will be identically distributed to the view $\mathsf{View}^{\mathsf{sm}}_{\mathsf{S}_{\mathsf{E}},\mathcal{A}}(\kappa)$ generated by the simulator S_{E}.

Hybrid Hyb_1: Hybrid 1 proceeds identically as the real game, except that in the challenge ciphertext the proof π^* is simulated. It immediately follows from the (composable) zero-knowledge property of Π, that Hyb_1 is indistinguishable from the real game (where the proof is not simulated).

Hybrid Hyb_2: Hybrid 2 proceeds as Hybrid 1, except that all the pre- and post-challenge decryption (with respect to original secret key) and tampering queries are answered by running the extractor Ext.

[8] Recall that, corresponding to each tampered key, $\mathcal{A}$ can make polynomially many decryption queries.
[9] Note that the challenger for the entropic restricted game expects a decryption query of the form (m, r).

It follows from the extractability property of the one-time strong tSE-NIZK that Hyb_1 and Hyb_2 are indistinguishable. This is true because the adversary only gets to see a single simulated argument of a *true* statement (pk, c'^*), and therefore cannot produce any new statement-argument pair (c'_i, π_i) for which the argument π_i verifies, but the extractor fails to extract the correct m_i.

Hybrid Hyb_3: Hybrid 3 proceeds as Hybrid 2, except that Hybrid 3 uses the entropic simulator S_{ER} to generate the ciphertext c^*. It follows from the semantic property of ER-BLT in the presence of pre- and post-challenge leakage and tampering queries that the view of the adversary in both Hyb_2 and Hyb_3 are computationally indistinguishable, even given the message m^*, leakage and answer to tampering queries.

Note that the description of Hybrid 3 is identical to the working of the simulator S_{E}. Thus we have shown that $\mathsf{View}^{\mathsf{sm}}_{\mathsf{S}_{\mathsf{E}},\mathcal{A}}(\kappa)$, i.e the view of the adversary $\mathcal{A}$ in the simulated game is computationally indistinguishable from the real game (even given the message m^*). We now show that the message m^* retains high min-entropy, given the simulated view. We construct an adversary $\mathcal{A}^{\mathcal{ER}}$ against the entropic restricted scheme ER-BLT. $\mathcal{A}^{\mathcal{ER}}$ internally invokes $\mathcal{A}$, and proceeds identically to S_{E}, except that it interacts with the entropic simulator S_{ER} to generate the challenge ciphertext c'^*, and to answer leakage and tampering queries. Hence, the view of adversary $\mathcal{A}^{\mathcal{ER}}$ consists of the view of adversary $\mathcal{A}$ and some additional information Γ, which consist of the trapdoors of the NIZK and the internal randomness used to create the proof π^*. So, we have:

$$\widetilde{\mathrm{H}}_\infty\big(m^* | \mathsf{View}^{\mathsf{sm}}_{\mathsf{S}_{\mathsf{ER}},\mathcal{A}^{\mathcal{ER}}}(\kappa)\big) = \widetilde{\mathrm{H}}_\infty\big(m^* | \mathsf{View}^{\mathsf{sm}}_{\mathsf{S}_{\mathsf{E}},\mathcal{A}}(\kappa), \Gamma\big) = \widetilde{\mathrm{H}}_\infty\big(m^* | \mathsf{View}^{\mathsf{sm}}_{\mathsf{S}_{\mathsf{E}},\mathcal{A}}(\kappa)\big) \geq k - \lambda_{\mathsf{post}} - t_{\mathsf{post}} \log p.$$

where the second equation follows from the fact that m^* and Γ are independent conditioned on $\mathsf{View}^{\mathsf{sm}}_{\mathsf{S}_{\mathsf{E}},\mathcal{A}}(\kappa)$. The last equation follows from the entropic property of ER-BLT. This completes the proof of Theorem 2. □

C Security model for split-state IND-CCA-BLT secure PKE

In this section, we give the detailed security model for IND-CCA-BLT secure PKE tolerating post-challenge leakage and tampering attacks.

Definition 13. (Split state IND-CCA-BLT secure PKE). A 2-split state IND-CCA-BLT secure PKE scheme $\mathcal{BLT} = (\mathcal{BLT}.\mathsf{Setup}, \mathcal{BLT}.\mathsf{Gen}, \mathcal{BLT}.\mathsf{Enc}, \mathcal{BLT}.\mathsf{Dec})$ consists of the following algorithms:

- $\mathcal{BLT}.\mathsf{Setup}(1^\kappa)$*: The setup algorithm takes as input the security parameter, and output the public parameters* params*, which is taken as input by all the algorithms.*
- $\mathcal{BLT}.\mathsf{Gen}(\mathsf{params})$*: The key generation algorithm comprises of two subroutines namely,* $\mathcal{BLT}.\mathsf{Gen}_1$ *and* $\mathcal{BLT}.\mathsf{Gen}_2$*. The subroutine* $\mathcal{BLT}.\mathsf{Gen}_i$ $(i \in \{1,2\})$ *generates the* i^{th} *public-secret key pair, i.e,* $(pk'_i, sk'_i) \leftarrow \mathcal{BLT}.\mathsf{Gen}_i(\mathsf{params}, r_i)$ *where* $r_i \in \{0,1\}^*$*. The public key consists of the pair* $pk' = (pk'_1, pk'_2)$ *and the secret key consists of the pair* $sk' = (sk'_1, sk'_2)$*.*

– $\mathcal{BLT}.\mathsf{Enc}_{pk'}(m)$*: The* (randomized) *encryption algorithm takes as input a message* m*, a public key* $pk' = (pk'_1, pk'_2)$*, and outputs a ciphertext* C*.*

– $\mathcal{BLT}.\mathsf{Dec}(C, sk' = (sk'_1, sk'_2))$*: The decryption consists of two partial decryption subroutines* $\mathcal{BLT}.\mathsf{Dec}_1$*,* $\mathcal{BLT}.\mathsf{Dec}_2$*, and a combining subroutine* $\mathcal{BLT}.\mathsf{Comb}$*. The decryption subroutine* $\mathcal{BLT}.\mathsf{Dec}_i$ $(i \in \{1,2\})$ *takes as input the ciphertext* C*, the secret key split* sk'_i *and outputs a partial decryption* t'_i*, i.e.,* $t'_i \leftarrow \mathcal{BLT}.\mathsf{Dec}_i(C, sk'_i)$*. Finally,* $\mathcal{BLT}.\mathsf{Comb}$ *takes the ciphertext* C *and the pair* (t'_1, t'_2) *to recover the plaintext* m*, i.e.,* $m \leftarrow \mathcal{BLT}.\mathsf{Comb}(C, t' = (t'_1, t'_2))$*.*

We want the usual *correctness* requirement to hold for $\mathcal{BLT}$, i.e., $\forall \mathsf{params} \leftarrow \mathcal{BLT}.\mathsf{Setup}(1^\kappa)$, $(pk'_i, sk'_i) \leftarrow \mathcal{BLT}.\mathsf{Gen}_i(\mathsf{params})\,(i \in \{1,2\}), \forall m \in \mathcal{M}$, we require that $\mathcal{BLT}.\mathsf{Dec}\big(sk' = (sk'_1, sk'_2), C = \mathcal{BLT}.\mathsf{Enc}_{pk'}(m)\big) = m$ holds with probability 1.

We now define the notion of CCA security of PKE schemes in the presence of after-the-fact split-state memory leakage and tampering attacks.

Definition 14. (Post-Challenge IND-CCA-BLT security in split-state) *Let* $\kappa \in \mathbb{N}$ *be the security parameter. Let* $\lambda_{\mathsf{pre}}(\kappa)$ *and* $\lambda_{\mathsf{post}}(\kappa)$ *be the upper bound on the amounts of memory leakage before and after the challenge phase respectively. Also, let* $t_{\mathsf{pre}}(\kappa)$ *and* $t_{\mathsf{post}}(\kappa)$ *be the bounds on the number of pre- and post-challenge tampering queries asked by the adversary before and after the challenge phase respectively. A* 2*-split-state PKE scheme* $\mathcal{BLT} = (\mathcal{BLT}.\mathsf{Setup}, \mathcal{BLT}.\mathsf{Gen},$
$\mathcal{BLT}.\mathsf{Enc}, \mathcal{BLT}.\mathsf{Dec})$ *is post-challenge* IND-CCA-$\big(k, (\lambda_{\mathsf{pre}}, \lambda_{\mathsf{post}}), (t_{\mathsf{pre}}, t_{\mathsf{post}})\big)$-BLT *secure if for all PPT adversaries* $\mathcal{B}$*, the advantage* $\mathsf{Adv}^{\mathsf{AFL\text{-}IND\text{-}CCA\text{-}BLT}}_{\mathcal{B},\mathcal{BLT}}(\kappa)$ *defined below is at most* $\frac{1}{2} + negl(\kappa)$*.*

1. **Key Generation:** The challenger chooses $r_1, r_2 \xleftarrow{\$} \{0,1\}^*$, and compute $(pk'_i, sk'_i) \leftarrow \mathcal{BLT}.\mathsf{Gen}_i(1^\kappa, r_i)$ *(*$i \in \{1,2\}$*)* and sends $pk' = (pk'_1, pk'_2)$ to the adversary, and keeps $sk' = (sk'_1, sk'_2)$ to itself. Also, it initializes two lists $\mathcal{L}^1_{\mathsf{pre}} = \mathcal{L}^2_{\mathsf{pre}} = 0$, where $\mathcal{L}^i_{\mathsf{pre}}$ denotes the random variable quantifying the amount of leakage from the i^{th} split sk'_i of the secret key sk' $(i \in \{1,2\})$.

2. **Pre-Challenge Leakage:** The adversary makes an arbitrary number of leakage queries $(f^{\mathsf{pre}}_{1,i}, f^{\mathsf{pre}}_{2,i})$ adaptively, where $f^{\mathsf{pre}}_{1,i}$ and $f^{\mathsf{pre}}_{2,i}$ acts independently on the secret key components sk'_1 and sk'_2 respectively. Upon receiving the i-th leakage query the challenger sends back $(f^{\mathsf{pre}}_{1,i}(sk'_1), f^{\mathsf{pre}}_{2,i}(sk'_2))$, provided $\mathcal{L}^1_{\mathsf{pre}} + |f^{\mathsf{pre}}_{1,i}(sk'_1)| \leq \lambda_{\mathsf{pre}}(\kappa)$ and $\mathcal{L}^2_{\mathsf{post}} + |f^{\mathsf{pre}}_{2,i}(sk'_2)| \leq \lambda_{\mathsf{pre}}(\kappa)$. It updates $\mathcal{L}^1_{\mathsf{pre}} = \mathcal{L}^1_{\mathsf{pre}} + |f^{\mathsf{pre}}_{1,i}(sk'_1)|$, and $\mathcal{L}^2_{\mathsf{post}} = \mathcal{L}^2_{\mathsf{post}} + |f^{\mathsf{pre}}_{2,i}(sk'_2)|$.

3. **Pre-Challenge Tampering:** The adversary is allowed to make at most t_{pre} number of pre-challenge tampering queries $(T^{\mathsf{pre}}_{1,i}, T^{\mathsf{pre}}_{2,i})$ for $i \in [t_{\mathsf{pre}}]$, where $T^{\mathsf{pre}}_{1,i}$ and $T^{\mathsf{pre}}_{2,i}$ acts independently on the secret key components sk'_1 and sk'_2 respectively. In more detail, for each of the tampering query $T_i = (T^{\mathsf{pre}}_{1,i}, T^{\mathsf{pre}}_{2,i})$, the adversary $\mathcal{B}$ gets access to the tampered decryption oracles $\mathcal{BLT}.\mathsf{Dec}(\widetilde{sk}'_{1,\psi}, \cdot)$ and $\mathcal{BLT}.\mathsf{Dec}(\widetilde{sk}'_{2,\psi}, \cdot)$, where $\widetilde{sk}'_{j,\psi} = T^{\mathsf{pre}}_{j,\psi}(sk'_j)$ (where $1 \leq \psi \leq i$, and $j \in \{1,2\}$). 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 queries on the decryption oracles are polynomial.

Note that, when $(T^{\mathsf{pre}}_{1,\psi}(sk'_1), T^{\mathsf{pre}}_{2,\psi}(sk'_2)) = (sk_1, sk_2)$, $\mathcal{B}$ gets access to the (normal) decryption oracle in the pre-challenge phase.

4. **Challenge:** In this phase, $\mathcal{B}$ gives two challenge messages m_0 and m_1, and the the challenger chooses $b \xleftarrow{\$} \{0,1\}$, computes $C^* = \mathcal{BLT}.\mathsf{Enc}_{pk}(m_b)$ and gives it to $\mathcal{B}$.
5. **Post-Challenge Leakage:** The adversary makes an arbitrary number of leakage queries $(f^{\mathsf{post}}_{1,j}, f^{\mathsf{post}}_{2,j})$ adaptively, where $f^{\mathsf{post}}_{1,j}$ and $f^{\mathsf{post}}_{2,j}$ act independently on the secret key components sk'_1 and sk'_2 respectively. Upon receiving the j^{th} leakage query, the challenger sends back $(f^{\mathsf{post}}_{1,j}(sk'_1), f^{\mathsf{post}}_{2,j}(sk'_2))$, provided $\mathcal{L}^1_{\mathsf{post}} + |f^{\mathsf{post}}_{1,j}(sk'_1)| \leq \lambda_{\mathsf{post}}(\kappa)$ and $\mathcal{L}^2_{\mathsf{post}} + |f^{\mathsf{post}}_{2,j}(sk'_2)| \leq \lambda_{\mathsf{post}}(\kappa)$. It updates $\mathcal{L}^1_{\mathsf{post}} = \mathcal{L}^1_{\mathsf{post}} + |f^{\mathsf{post}}_{1,j}(sk'_1)|$, and $\mathcal{L}^2_{\mathsf{post}} = \mathcal{L}^2_{\mathsf{post}} + |f^{\mathsf{post}}_{2,j}(sk'_2)|$.
6. **Post-Challenge Tampering:** The adversary is allowed to make at most t_{post} number of pre-challenge tampering queries $(T^{\mathsf{post}}_{1,j}, T^{\mathsf{post}}_{2,j})$ for $j \in [t_{\mathsf{post}}]$, where $T^{\mathsf{post}}_{1,j}$ and $T^{\mathsf{post}}_{2,j}$ act independently on the secret key components sk'_1 and sk'_2 respectively. In more detail, for each of the tampering query $T_j = (T^{\mathsf{post}}_{1,j}, T^{\mathsf{post}}_{2,j})$, the adversary $\mathcal{B}$ is allowed to ask polynomial number of decryption queries, in which case $\mathcal{B}$ gets access to the tampered decryption oracles $\mathcal{BLT}.\mathsf{Dec}(\widetilde{sk}'_{1,\varsigma}, .)$ and $\mathcal{BLT}.\mathsf{Dec}(\widetilde{sk}'_{2,\varsigma}, .)$ respectively $(1 \leq \varsigma \leq j)$, as before. However, in the post-challenge phase, an additional restriction is imposed on the tampering functions T_ς: When the adversary asks tampering functions T_ς, and gets access to the decryption oracles $\mathcal{BLT}.\mathsf{Dec}(\widetilde{sk}'_{1,\varsigma}, C^*)$ and $\mathcal{BLT}.\mathsf{Dec}(\widetilde{sk}'_{2,\varsigma}, C^*)$ with respect to the challenge ciphertext C^*, it should hold that $\widetilde{sk}'_{1,\varsigma} \neq sk_1$ and $\widetilde{sk}'_{2,\varsigma} \neq sk_2$.
7. **Guess:** Finally, the adversary outputs a bit b' for a guess of the bit b chosen the challenger. If $b' = b$, output 1, else output 0.

We define the advantage of the adversary $\mathcal{B}$ in the above experiment as:

$$\mathsf{Adv}^{\mathsf{AFL\text{-}IND\text{-}CCA\text{-}BLT}}_{\mathcal{B},\mathcal{BLT}}(\kappa) = \big|\mathsf{Pr}[b' = b] - \frac{1}{2}\big|.$$

D Proof of Theorem 5

Theorem 6. *The encryption scheme $\mathcal{BLT}$ is post-challenge* IND-CCA-$\big(k, (\lambda''_{\mathsf{pre}}, \lambda''_{\mathsf{post}}), (t''_{\mathsf{pre}}, t''_{\mathsf{post}})\big)$-BLT *secure as long as the parameters satisfies:*

$$\lambda''_{\mathsf{pre}} \leq \lambda_{\mathsf{pre}}{}', \quad \lambda''_{\mathsf{post}} \leq \lambda_{\mathsf{post}}{}' \quad \text{and} \quad t''_{\mathsf{pre}} \leq t'_{\mathsf{pre}}, \quad t''_{\mathsf{post}} \leq t'_{\mathsf{post}}.$$

Proof. We need to show that the advantage of any PPT adversary $\mathcal{B}$ in the AFL-IND-CCA-BLT secure game for the PKE scheme $\mathcal{BLT}$ is negligible. For this, we introduce an intermediate hybrid experiment Hyb_1, and show that if the adversary $\mathcal{B}$ can distinguish the real game from Hyb_1, then it can break the AFL-IND-CCA-BLT security of the KEM scheme $\mathcal{KEM}$. Finally, we show that the advantage of the adversary in Hyb_1 is upper bounded by the advantage of an adversary $\mathcal{B}_{\mathcal{SKE}}$ against the symmetric-key encryption scheme φ.

Hyb_1 : In this hybrid,the challenger proceeds as in the real game, except for two main differences: Firstly, the challenger generates the challenge ciphertext $c^* = (c_0^*, c_1^*)$ as $(c_0^*, k^*) \leftarrow \mathcal{KEM}.\mathsf{Encap}(pk')$, and encrypting the message m_b in c_1^* using a randomly chosen encapsulation key $k \in \{0,1\}^u$. Secondly, when the adversary submits a ciphertext of the form $c = (c_0^*, c_1)$ to the decryption oracle or the tampering oracle, if $c \neq c^*$, the challenger does not run the $\mathcal{KEM}.\mathsf{Decap}$ algorithm to obtain the encapsulated symmetric key; instead, it uses the key k to decrypt. Let us denote the adversary for $\mathcal{KEM}$ by $\mathcal{B}_{\mathcal{KEM}}$

Claim. $\left|\mathsf{Adv}^{\mathsf{AFL\text{-}IND\text{-}CCA\text{-}BLT}}_{\mathcal{B},\mathcal{BLT}}(\kappa) - \mathsf{Adv}^{\mathsf{AFL\text{-}IND\text{-}CCA\text{-}BLT}}_{\mathcal{B},\mathsf{Hyb}_1}(\kappa)\right| \leq \mathsf{negl}(\kappa)$

Proof. Suppose, for contradiction, there exists some polynomial $p(\kappa)$, and $\kappa \in \mathbb{N}$ such that the advantage of $\mathcal{B}$ in distinguishing the real game from Hybrid 1 is at least $1/p(\kappa)$. We then show that the adversary $\mathcal{B}_{\mathcal{KEM}}$ can break AFL-IND-CCA-BLT security of KEM with non-negligible advantage using $\mathcal{B}$ as a black-box. The adversary $\mathcal{B}_{\mathcal{KEM}}$ proceeds as shown below:

1. In the key generation phase $\mathcal{B}_{\mathcal{KEM}}$ receives as input the public key $pk' = (pk'_1, pk'_2)$ from the external challenger $\mathcal{C}$ of the KEM scheme. It then returns pk' to $\mathcal{B}$.
2. In the pre-challenge phase, when $\mathcal{B}$ makes leakage queries $f_i = (f^{\mathsf{pre}}_{1,i}, f^{\mathsf{pre}}_{2,i})$, $\mathcal{B}_{\mathcal{KEM}}$ forwards f_i to $\mathcal{C}$. It then gets the answer from $\mathcal{C}$ and forwards it to $\mathcal{B}$.
3. When $\mathcal{B}$ makes a tampering query $T_i = (T^{\mathsf{pre}}_{1,i}, T^{\mathsf{pre}}_{2,i})$, forward T_i to $\mathcal{C}$, and get the tampered key $\widetilde{k}_i$.
4. When $\mathcal{B}$ asks decryption queries $c = (c_0, c_1)$ with respect to the i^{th} tampered key (say), $\mathcal{B}_{\mathcal{KEM}}$ first checks if $i \in [t]$. If so, it checks if it has the i^{th} tampered key. If not, it makes the i^{th} tampering query to $\mathcal{C}$ to get $\widetilde{k}_i$. It then runs $\mathcal{SKE}.\mathsf{Dec}(\widetilde{k}_i, c_1)$ to return the resulting message to $\mathcal{B}$.
5. When $\mathcal{B}$ asks a decryption query $c = (c_0, c_1)$ with respect to the original secret key, $\mathcal{B}_{\mathcal{KEM}}$ forwards c_0 to the challenger $\mathcal{C}$, gets the answer k. It then runs $\mathcal{SKE}.\mathsf{Dec}(k, c_1)$, and returns back the resulting message to $\mathcal{B}$.
6. In the challenge phase, when $\mathcal{B}$ submits two messages m_0, m_1 of equal length, $\mathcal{B}_{\mathcal{KEM}}$ asks the external challenger $\mathcal{C}$ for a ciphertext-key pair (c_0^*, k^*). It then randomly chooses a bit b, and computes the ciphertext $c_1^* = \mathcal{SKE}.\mathsf{Enc}(k^*, m_b)$. It then sends $c^* = (c_0^*, c_1^*)$ to $\mathcal{B}$.
7. In the post-challenge phase, the leakage, tampering and decryption queries are handled identically as in pre-challenge phase, except that when $\mathcal{B}$ asks a decryption query on $c = (c_0^*, c_1^*)$, $\mathcal{B}_{\mathcal{KEM}}$ uses the key k^* to decrypt.
8. Finally $\mathcal{B}_{\mathcal{KEM}}$ outputs whatever $\mathcal{B}$ outputs.

For the analysis, note that, when the external challenger $\mathcal{C}$ generates a ciphertext-key pair using $(c_0^*, k^*) \leftarrow \mathcal{KEM}.\mathsf{Encap}(pk')$, $\mathcal{B}_{\mathcal{KEM}}$ acts identically as in real game. On the other hand, if $\mathcal{C}$ chooses the encapsulation key k^* uniformly at random, this corresponds to hybrid Hyb_1. Also, note that, since k^* is uniformly and randomly chosen from the distribution of encapsulation key space (bit strings of length u), the min-entropy of k^* even given the tampered keys $\widetilde{k} = (\widetilde{k}_1, \cdots, \widetilde{k}_t)$ is $\widetilde{\mathrm{H}}_\infty(k^* | (pk', c^*, \widetilde{k})) = \widetilde{\mathrm{H}}_\infty(k^*) = -\log(2^{-u}) = u$. Hence by Lemma 3, we have:

$$\widetilde{\mathrm{H}}_\infty(k^* | (pk', c^*, \mathsf{Dec}(\widetilde{k}_1, \cdot), \cdots, \mathsf{Dec}(\widetilde{k}_t, \cdot))) = \widetilde{\mathrm{H}}_\infty(k^* | (pk', c^*, \widetilde{k} = (\widetilde{k}_1, \cdots, \widetilde{k}_t))) = u$$

Thus, we get:

$$\mathsf{Adv}^{\mathsf{AFL\text{-}IND\text{-}CCA\text{-}BLT}}_{\mathcal{B}_{\mathcal{KEM}}, \mathcal{BLT}}(\kappa) = \frac{1}{2} \left| \mathsf{Adv}^{\mathsf{AFL\text{-}IND\text{-}CCA\text{-}BLT}}_{\mathcal{B}, \mathcal{BLT}}(\kappa) - \mathsf{Adv}^{\mathsf{AFL\text{-}IND\text{-}CCA\text{-}BLT}}_{\mathcal{B}, \mathsf{Hyb}_1}(\kappa) \right| \geq \frac{1}{2p(\kappa)}.$$

In the next claim we show that the advantage of any PPT adversary in $\mathbf{Hyb}_1$ is negligible.

Claim. $\mathsf{Adv}^{\mathsf{AFL\text{-}IND\text{-}CCA\text{-}BLT}}_{\mathcal{B}, \mathsf{Hyb}_1}(\kappa) \leq \mathsf{Adv}^{\mathcal{SKE}}_{\varphi, \mathcal{B}}(\kappa)$

Proof. We now describe an adversary $\mathcal{A}^{\mathcal{SKE}}$ for the symmetric key encryption scheme φ. $\mathcal{A}^{\mathcal{SKE}}$ proceeds identically as in Hyb_1, except that all of the symmetric key operations are forwarded to the external SKE challenger φ. In more details $\mathcal{A}^{\mathcal{SKE}}$ proceeds as follows:

1. In the key generation phase, $\mathcal{A}^{\mathcal{SKE}}$ generates two key pairs (pk_1, sk_1) and (pk_2, sk_2) by invoking the algorithm $\mathcal{KEM}.\mathsf{Gen}$. It then forwards $pk' = (pk_1, pk_2)$ to the adversary $\mathcal{B}$.
2. In the pre-challenge phase, when $\mathcal{B}$ makes leakage queries $f_i = (f^{\mathsf{pre}}_{1,i}, f^{\mathsf{pre}}_{2,i})$, $\mathcal{A}^{\mathcal{SKE}}$ computes $(f^{\mathsf{pre}}_{1,i}(sk_1), f^{\mathsf{pre}}_{1,i}(sk_2))$ and returns the answers to $\mathcal{B}$, as long as the leakage bounds are respected.
3. When $\mathcal{B}$ makes a tampering query $T_i = (T^{\mathsf{pre}}_{1,i}, T^{\mathsf{pre}}_{2,i})$, compute the tampered secret keys $\widetilde{sk}_i = (T^{\mathsf{pre}}_{1,i}(sk_1), T^{\mathsf{pre}}_{2,i}(sk_2)$. On input a decryption query $c = (c_0, c_1)$ under the tampered key $\widetilde{sk}_i$ (say), $\mathcal{A}^{\mathcal{SKE}}$ decrypts c_0 under $\widetilde{sk}_i$ to get an encapsulation key $\widetilde{k}_i$, and then it runs $\mathcal{SKE}.\mathsf{Dec}(\widetilde{k}_i, c_1)$, returning the result to $\mathcal{B}$.
4. When $\mathcal{B}$ asks a decryption query $c = (c_0, c_1)$ with respect to the original secret key sk', $\mathcal{A}^{\mathcal{SKE}}$ decrypts c_0 itself to get an encapsulated key k, and then runs $\mathcal{SKE}.\mathsf{Dec}(k, c_1)$, returning the result to $\mathcal{B}$.
5. In the challenge phase, when $\mathcal{B}$ submits two messages m_0, m_1, $\mathcal{A}^{\mathcal{SKE}}$ generates a ciphertext-key pair (c^*_0, k). It then submits m_0, m_1 to the challenger of φ and get back the ciphertext c_1* (under some key k^*). $\mathcal{A}^{\mathcal{SKE}}$ then returns $c^* = (c^*_0, c^*_1)$ to $\mathcal{B}$.
6. In the post-challenge phase, the leakage, tampering and decryption queries are handled identically as in pre-challenge phase, except that if $\mathcal{B}$ asks a decryption query $c = (c^*_0, c_1) \neq c^*$, $\mathcal{A}^{\mathcal{SKE}}$ asks the challenger of φ to decrypt c_1.
7. Finally, $\mathcal{A}^{\mathcal{SKE}}$ outputs whatever $\mathcal{B}$ outputs.

From the above simulation, we get that $\mathsf{Adv}^{\mathsf{AFL\text{-}IND\text{-}CCA\text{-}BLT}}_{\mathcal{B}, \mathsf{Hyb}_1}(\kappa) \leq \mathsf{Adv}^{\mathcal{SKE}}_{\varphi, \mathcal{B}}(\kappa)$. Since φ is a CCA-secure SKE scheme, it follows that $\mathsf{Adv}^{\mathcal{SKE}}_{\varphi, \mathcal{B}}(\kappa)$ is negligible (in κ). Thus, $\mathsf{Adv}^{\mathsf{AFL\text{-}IND\text{-}CCA\text{-}BLT}}_{\mathcal{B}, \mathsf{Hyb}_1}(\kappa)$ is also negligible in κ.

Finally, combining the above two claims, we get the proof of Theorem 5. □