# Targeted Lossy Functions and Applications

Anonymous Submission

No Institute Given

**Abstract.** Lossy trapdoor functions, introduced by Peikert and Waters (STOC '08), can be initialized in one of two indistinguishable modes: in injective mode, the function preserves all information about its input, and can be efficiently inverted given a trapdoor, while in lossy mode, the function loses some information about its input. Such functions have found countless applications in cryptography, and can be constructed from a variety of Cryptomania assumptions. In this work, we introduce *targeted lossy functions (TLFs)*, which relax lossy trapdoor functions along two orthogonal dimensions. Firstly, they do not require an inversion trapdoor in injective mode. Secondly, the lossy mode of the function is initialized with some target input, and the function is only required to lose information about this particular target. The injective and lossy modes should be indistinguishable even given the target. We construct TLFs from Minicrypt assumptions, namely, injective pseudorandom generators, or even one-way functions under a natural relaxation of injectivity. We then generalize TLFs to incorporate *branches*, and construct *all-injective-but-one* and *all-lossy-but-one* variants. We show a wide variety of applications of targeted lossy functions. In several cases, we get the first Minicrypt constructions of primitives that were previously only known under Cryptomania assumptions. Our applications include:

- *Pseudo-entropy functions* from one-way functions.
- Deterministic leakage-resilient message-authentication codes and improved leakage-resilient symmetric-key encryption from one-way functions.
- Extractors for *extractor-dependent sources* from one-way functions.
- Selective-opening secure symmetric-key encryption from one-way functions.
- A new construction of CCA PKE from (exponentially secure) trapdoor functions and injective pseudorandom generators.

We also discuss a fascinating connection to distributed point functions.

## 1 Introduction

Lossy trapdoor functions, introduced by Peikert and Waters [PW08], are a fundamental cryptographic tool and have found countless applications in many areas of cryptography. They can be constructed from a wide variety of specific Cryptomania assumptions. In this work, we introduce a relaxation of lossy trapdoor functions that we call *targeted-lossy functions* (TLFs), and show how to instantiate them using Minicrypt assumptions. We then provide applications of TLFs

to a diverse set of problems. For several of these problems, we get the first solutions under one-way functions, where previously only solutions under specific Cryptomania assumptions were known.

*Lossy Trapdoor Functions.* Lossy trapdoor functions consist of a function family $F_{\mathsf{fk}}(\cdot)$ indexed by a public *function key* $\mathsf{fk}$. The function key $\mathsf{fk}$ can be generated in one of two modes. In *injective* mode, the function $F_{\mathsf{fk}}(\cdot)$ is injective, and therefore each output $y = F_{\mathsf{fk}}(x)$ uniquely determines the input $x$. Furthermore, the public function key $\mathsf{fk}$ is generated together with a secret trapdoor $\mathsf{td}$ that allows one to efficiently invert the function and recover the input $x$ from the output $y = F_{\mathsf{fk}}(x)$. In *lossy* mode, the output $y = F_{\mathsf{fk}}(x)$ loses some information about the input $x$. This is captured by defining a lossiness parameter $\ell$ and requiring that the size of the image of $F_{\mathsf{fk}}$ is at most a $\frac{1}{2^\ell}$ fraction of the size of the domain. In particular, this implies that when $x$ is uniformly random over its domain, then the conditional entropy[1] of $x$ given $F_{\mathsf{fk}}(x)$ is at least $\ell$ bits, meaning that this information about $x$ is lost by $F_{\mathsf{fk}}(x)$. The two modes should be *computationally indistinguishable*: given $\mathsf{fk}$, one cannot tell if it was generated in injective mode or lossy mode.

Since their introduction, lossy trapdoor functions have turned out to be incredibly versatile tool and have quickly become an integral part of our cryptgoraphic tool-set. They have found countless and varied applications, including to CCA security, trapdoor functions with many hard-core bits, collision-resistant hash functions, and oblivious transfer [PW08], deterministic encryption [BFO08], analyzing OAEP [KOS10], hedged public-key encryption with bad randomnes [BBN+09], selective opening security [BHY09], pseudo-entropy functions [BHK11], point-function obsuscation [Zha16], computational extractors [DVW20, GKK20], incompressible encodings [MW20], etc.

Lossy trapdoor functions are known to imply public-key encryption, making them a *Cryptomania* primitive. We currently know how to construct lossy trapdoor functions under most (but not all) concrete Cryptomania assumptions, such as DDH, LWE, Quadratic Residuosity (QR), Decision-Composite Residuosity (DCR), and Phi-hiding [PW08, KOS10, FGK+13], but not e.g., factoring, RSA, or the (low noise) LPN assumption.

*Targeted-Lossy Functions.* In this work, we introduce a relaxation of lossy trapdoor functions, that we call *targeted-lossy functions (TLFs)*, with the goal of constructing them under weaker assumptions. TLFs relax the notion of lossy trapdoor functions along two orthogonal dimensions:

– *No inversion trapdoor in injective mode.* When we generate $\mathsf{fk}$ in injective mode, we now *only* require that the function $F_{\mathsf{fk}}(\cdot)$ is injective, but we no longer require there to be a trapdoor $\mathsf{td}$ that allows us to efficiently invert it.

[1] Throughout the introduction, entropy refers to min-entropy, and conditional entropy refers to average-case conditional min-entropy [DORS08].

– *Targeted Lossiness.* When we generate $\mathsf{fk}$ in lossy mode, we are now also given a target input $x^*$. We require that, when the target $x^*$ is uniformly random over the domain and $\mathsf{fk}$ is chosen in lossy mode with the target $x^*$, then the pair $\mathsf{fk}, F_{\mathsf{fk}}(x^*)$ loses $\ell$ bits of information about $x^*$. Formally, the conditional entropy of $x^*$ given $(\mathsf{fk}, F_{\mathsf{fk}}(x^*))$ should be at least $\ell$ bits.

The two modes should be computationally indistinguishable even given the potential target $x^*$. In other words, given the pair $(\mathsf{fk}, x^*)$, one cannot distinguish whether $\mathsf{fk}$ was chosen in injective mode and independently of $x^*$ or in lossy mode with $x^*$ as the target.

Notice that the first relaxation already appears to take us out of Cryptomania – without a trapdoor, there is no obvious way to use this primitive to construct public-key encryption. Moreover, this relaxation was considered in prior works (e.g., [BHK11, DVW20]), and is already known to have interesting applications. Unfortunately, there has been no progress towards achieving this relaxation on its own under any Minicrypt assumption, or even under any assumption that doesn't already imply the full notion of lossy trapdoor functions. This motivates us to consider this relaxations in conjunction with our second relaxation.

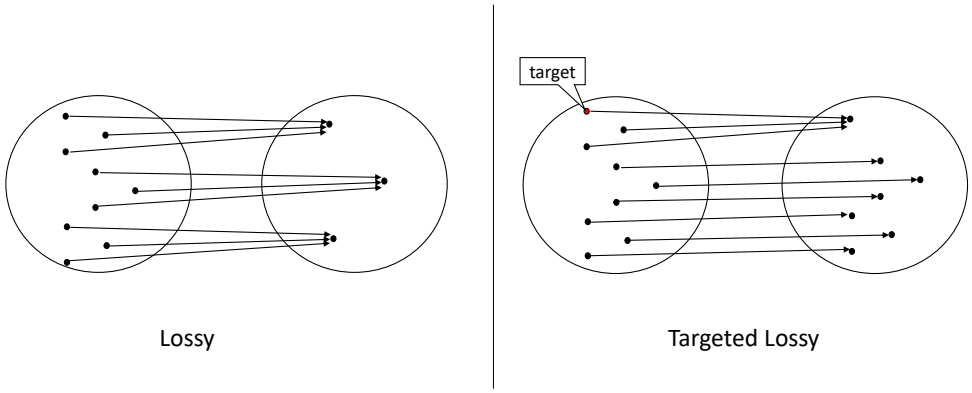


**Fig. 1.** Lossy vs Targeted-Lossy

The second relaxation to targeted lossiness is substantially weaker than standard lossiness. For example, targeted lossiness with parameter $\ell$ could be achieved by choosing a lossy function key $\mathsf{fk}$ for some target $x^*$ where $y = F_{\mathsf{fk}}(x^*)$ has $2^\ell$ pre-images in the set $S = \{x \ : \ F_{\mathsf{fk}}(x) = y\}$, but for every $x \notin S$, the value $F_{\mathsf{fk}}(x)$ has a unique pre-image $x$. Such a function would *not* be lossy in the standard sense. For example, if the domain of the function is $\{0,1\}^n$ with $n = 2\ell$, then, from the point of view of standard lossiness, the function only has negligible information loss $\ell' = O(2^{-\ell})$, even though its targeted-lossiness $\ell$ can

be an arbitrarily large polynomial.[2] This distinction is illustrated in Figure 1. Despite the large difference between the notions, we show that targeted lossiness suffices in many applications in place of standard lossiness.[3]

*TLFs with Branches/Tags.* We also introduce TLFs with branches/tags, analogously to prior notions of branches for lossy trapdoor functions [PW08]. In this setting, a single function key $\mathsf{fk}$ defines an entire family of functions $F_{\mathsf{fk},\mathsf{tag}}(\cdot)$ with various branches indexed by $\mathsf{tag}$. We can sample the function key $\mathsf{fk}$ with a special branch $\mathsf{tag}^*$ and a target value $x^*$, and we require that the pair $(\mathsf{fk}, x^*)$ computationally hides $\mathsf{tag}^*$. We define two main variants of this notion, depending on whether the special branch is lossy or injective.

In a targeted *all-injective-but-one* (T-A**I**BO) family, the special branch $\mathsf{tag}^*$ is targeted-lossy and all other branches are injective. In particular, for all $\mathsf{tag} \neq \mathsf{tag}^*$, the function $F_{\mathsf{fk},\mathsf{tag}}$ is injective, while $F_{\mathsf{fk},\mathsf{tag}}(x^*)$ loses $\ell$ bits of information about the target $x^*$. This notion is most directly analogous to the way branches were defined for standard lossy trapdoor functions of [PW08].

In a targeted *all-lossy-but-one* (T-A**L**BO) family, the function $F_{\mathsf{fk},\mathsf{tag}^*}$ is injective on the special branch $\mathsf{tag}^*$, while all other branches $F_{\mathsf{fk},\mathsf{tag}}$ are *cumulatively* targeted-lossy. In particular, the cumulative outputs of all lossy branches $\{F_{\mathsf{fk},\mathsf{tag}}(x^*) \ : \ \mathsf{tag} \neq \mathsf{tag}^*\}$ must lose $\ell$-bits of information about the target $x^*$. An analogous notion of branches for the case of lossy trapdoor functions was previously considered in [CPW20], and even a relaxed version without trapdoors (but without the relaxation to targeted lossiness) was considered implicitly in [BHK11, GKK20] and explicitly in [DVW20]. All prior constructions relied on Cryptomania assumptions.

*Relaxing Injectivity.* It turns out that we can also relax the injectivity requirement of TLFs further, while sufficing for most of our applications. When we choose $\mathsf{fk}$ in injective mode, instead of requiring that $F_{\mathsf{fk}}(x)$ uniquely determines $x$, we only require that it uniquely determines some property of $x$, modeled as an arbitrary function $P(x)$. In this case, we also require that when $\mathsf{fk}$ is in lossy mode for the target value $x^*$, then $F_{\mathsf{fk}}(x^*)$ loses $\ell$ bits of information about the same property $P(x^*)$. We do not care what the property $P$ is, as long as some such property exists. We can define T-A**I**BOs and T-A**L**BOs with relaxed injectivity analogously, but even allow the property $P$ to depend on $\mathsf{fk}$. In the case of T-A**L**BOs, by setting the property $P(x) = F_{\mathsf{fk},\mathsf{tag}^*}(x)$, this relaxed injectivity requirement is equivalent to insisting that the cumulative outputs of all lossy

[2] The function has an image of size $2^n - 2^\ell + 1$, which we can write as $\frac{1}{2^{\ell'}}2^n$ for $\ell' = O(2^{-\ell})$.

[3] We can also consider the second relaxation to targeted lossiness on its own without making the first relaxation (i.e., by still insisting on an inversion trapdoor in injective mode). In that case, the resulting notion would still be a Cryptomania primitive. Interestingly, this notion was considered informally in [GGH19], where it was constructed under the CDH assumption, which is not known to imply standard lossy trapdoor functions.

branches $\{F_{\mathsf{fk},\mathsf{tag}}(x^*) \ : \ \mathsf{tag} \neq \mathsf{tag}^*\}$ should lose $\ell$ bits of information about the output of the "injective" branch $F_{\mathsf{fk},\mathsf{tag}^*}(x^*)$.

### 1.1 Our Results

We construct targeted lossy functions (TLFs) from injective pseudorandom generators (PRGs). We also generalize our construction to targeted all-injective-but-one functions (T-A**I**BOs) under the same assumption. For all-lossy-but-one functions (T-A**L**BOs), we need a stronger "doubly injective" PRG $G(x) = (y_0, y_1)$, whose output consists of two halves $y_0, y_1$, and the PRG is injective on each half individually (i.e., either one of $y_0, y_1$ uniquely determines $x$). We also construct TLFs, T-A**I**BOs and T-A**L**BOs with relaxed injectivity from just one-way functions. In all cases, we start with a basic construction that only achieves lossiness of $\ell = 1$ bits, but can then amplify lossiness via parallel repetition to get to an arbitrary polynomial $\ell$. (However, the lossiness *rate* of our constructions, defined as $\ell/n$ where $n$ is the domain size, is stuck at $1/\lambda$, where $\lambda$ is the security parameter — achieving a higher lossiness rate in Minicrypt is a fascinating open problem.)

*Application: Pseudo-entropy Functions.* The work of Braverman, Hassidim, and Kalai [BHK11] introduced the notion of a *pseudo-entropy function (PEF)*, and constructed them under the DDH assumption. A PEF $f_k(x)$ has a secret key $k$ and takes as inputs values $x$. The requirement is that for any a-priori chosen input $x^*$, we can indistinguishibly select the key $k$ so that $f_k(x^*)$ has $\ell$ bits of true statistical entropy even given $f_k(x)$ for all inputs $x \neq x^*$. We observe that PEFs follow immediately from T-A**L**BOs with relaxed injectivity, and therefore get a construction of PEFs from one-way functions. The amount of entropy $\ell$ in our construction can be set to an arbitrarily large polynomial. (On the other hand, the entropy *rate* of our construction, defined as $\ell/n$ where $n$ is the key size, is stuck at $1/\lambda$. This is in contrast to the construction of [BHK11], which achieved an entropy rate of $1 - o(1)$ under DDH.)

*Application: Leakage-Resilience. Leakage-resilient cryptography* aims to preserve security even if an adversary can get some partial leakage on the secret key. We consider the setting of memory leakage [AGV09,ADW09,NS09,HLWW13], where an adversary can learn any efficiently computable function of the secret key, as long as the number of leaked bits is bounded by some parameter $\ell$. As shown by [BHK11], pseudo-entropy functions (PEFs) are useful for leakage-resilient symmetric-key cryptography and were previously used to construct (selectively secure) deterministic leakage-resilient MACs under DDH. By using our new construction of PEFs from one-way functions, we get the first (selectively secure) deterministic leakage-resilient MACs from just one-way functions. The amount of leakage $\ell$ that we can tolerate can be set to an arbitrarily large polynomial. (However, the length of the key grows depending on $\ell$ and the leakage *rate* of our construction, defined as $\ell/n$ where $n$ is the key size, is stuck at $1/\lambda$. This is in contrast to the construction of [BHK11], which achieved a leakage rate of $1-o(1)$

under DDH.) We can also use a similar technique to construct leakage-resilient CPA-secure symmetric-key encryption from one-way functions.

We note that a prior work of [HLWW13] constructed leakage-resilient symmetric-key primitives, including CPA-secure symmetric-key encryption and (adaptively secure) MACs from one-way functions. The amount of leakage and the leakage rate are the same as in our construction. However, the MACs in the prior work were inherently randomized, while in this work we get deterministic MACs. This is especially crucial in the context of leakage-resilience since, in a randomized construction, leakage that occurs during a computation may also depend on the randomness of the computation in addition to the secret key, but such leakage was not analyzed by the prior work (and indeed, the proof there would fail). Furthermore, our MAC has a smaller tag size: the ratio of leakage to tag size is $(1 - o(1))$ in our construction while it is $1/\lambda$ in the prior work. For the case of CPA-secure symmetric-key encryption, in the prior work the ciphertext size grew linearly with the leakage bound $\ell$, while in our work, only the secret key size grows with the leakage bound $\ell$, but the ciphertext size just has a minimal $O(\lambda)$ additive overhead on top of the message length. For both MACs and symmetric-key encryption, our constructions are substantially different from those of [HLWW13].

*Application: Extractor-Dependent Sources.* The work of Dodis, Vaikuntanathan and Wichs [DVW20], which we will refer to as DVW, defined the notion of (computational) extractors for extractor-dependent sources. The goal is to extract nearly uniform randomness $R$ from an arbitrary source of randomness $X$ that has some sufficient entropy. Classical results show this to be possible using a *seeded* randomness extractor $R = \mathsf{Ext}(X; S)$, which relies on a public random seed $S$. As long as the source $X$ is independent of the seed $S$, the output $R$ is nearly uniform even given $S$. Usually, we think of the source $X$ as coming from nature and therefore consider it to be worst-case but not adversarial – this is used to justify its independence from $S$. DVW considered a setting where the seed $S$ is repeatedly used to extract randomness from nature and may therefore impact nature itself (e.g., we use timing of interrupts to derive entropy, but the interrupts may depend on processes that may themselves rely on extracted outputs). They model this by assuming that the source which produces $X$ can depend on oracle access to the extractor $\mathsf{Ext}(\cdot; S)$, but is independent of the seed $S$ otherwise, and they refer to such sources as *extractor-dependent sources*. DVW showed that extractors for extractor-dependent sources cannot exist unconditionally and at least imply one-way functions. They also distinguished between two scenarios, depending on whether the source can output some additional correlated *auxiliary information* $\mathsf{AUX}$ in addition to the sample $X$, as long as it preserves the entropy of $X$. The setting with auxiliary information is considered more realistic. As their main results, DVW show how to construct extractors for extractor-dependent sources in the setting without auxiliary information from sub-exponentially secure one-way functions, and in the setting with auxiliary information from a wide range of Cryptomania assumptions such as DDH, DLIN, LWE or DCR. They also gave some evidence that it would be difficult to con-

struct such extractors from simple Minicrypt primitives, by showing that a large class of constructions — ones where seeing the outputs of the extractor on many inputs uniquely determines the seed — cannot be proven secure via a black box reduction.

Despite the above negative result, in this work we construct extractors for extractor-dependent sources, even in the setting with auxiliary information, from standard one-way functions! Our construction does not require sub-exponential security, is entirely black-box in the one-way function, and achieves the same parameters as the prior constructions from Cryptomania assumptions. We circumvent the negative result of DVW by using a construction that lies outside of the class considered there — by relying on lossiness, we ensure that many outputs don't uniquely determine the seed — yet can still be instantiated in Minicrypt. Our main technique is to adapt a construction of DVW, which relied on all-lossy-but-one functions (without a trapdoor), and adapt it to only rely on targeted all-lossy-but-one functions.

*Applications: Selective Opening Security.* We also apply TLFs to the problem of selective opening security [DNRS99, BHY09] for symmetric-key encryption. A selective opening attack considers a scenario where an adversary sees a large number of ciphertexts and adaptively asks to "open" some subset of them; we would like to argue that the adversary does not learn anything about the messages encrypted in the remaining ciphertexts. An opening could correspond to seeing the encryption randomness or, if all the ciphertexts are encrypted under different keys, then seeing the corresponding secret keys. Surprisingly, selective opening security does not follow generically from standard encryption security [BDWY12, HR14, HRW16]. On the other hand, we have constructions of selective-opening secure public-key encryption for both randomness-opening and key-opening under many specific public-key assumptions [BHY09,FHKW10, HLOV11, Hof12, HPW15]. However, the problem does not appear to have been studied in the symmetric-key setting. One piece of good news is that symmetric-key encryption schemes are often "public coin", meaning that the encryption randomness is sent in the clear as part of the ciphertext. Such schemes are automatically secure against selective randomness-opening attacks, since the randomness is available to the adversary for free! Therefore, we focus on the case of selective key-opening attacks. We consider a setting where $n$ secret keys $k_1, \ldots, k_n$ are chosen uniformly at random and the adversary is given a CPA oracle for each of these keys. In addition, the adversary gets $n$ challenge ciphertexts, one under each key. The adversary can adaptively choose to open some arbitrary subset of the $n$ ciphertexts and we want to argue that the messages encrypted in the remaining ciphertexts remain hidden. Formalizing this requires some care and we naturally adapt the simulation-based definition of selective security from the public-key setting. We show how to construct such selectively secure symmetric-key encryption from one-way functions.

*Application: CCA Encryption from Injective Trapdoor Functions* The recent work of [HKW20] gave a black-box construction of CCA-secure public-key en-

cryption from any injective trapdoor function. In this work, we give a completely different construction using targeted all-injective-but-one functions (T-A**I**BOs). As our final result, we get CCA-secure public-key encryption from any injective trapdoor function with a very high (strongly exponential) level of security and an injective pseudorandom generator. While our end-result is strictly worse than [HKW20] in terms of the assumptions, our construction is conceptually simple and we hope it may point to further applications and/or improvements.

### 1.2 Our Techniques

*Basic Construction.* Our basic construction of targeted lossy functions (TLFs) with lossiness $\ell = 1$ is extremely simple. Let $G : \{0,1\}^\lambda \to \{0,1\}^{3\lambda+1}$ be an injective pseudorandom generator (PRG) and let $\mathcal{H} = \{h : \{0,1\}^{3\lambda+1} \to \{0,1\}^{3\lambda}\}$ be a universal hash function family, where $\lambda$ is the security parameter. We define the function family $F_{\mathsf{fk}} : \{0,1\}^\lambda \to \{0,1\}^{3\lambda}$ via $F_{\mathsf{fk}}(x) = h(G(x))$ where $\mathsf{fk} = h \in \mathcal{H}$.

The above parameters ensure that if we choose $\mathsf{fk} = h$ randomly, then the function $F_{\mathsf{fk}}$ is injective with overwhelming probability. In particular, for any $x_0 \neq x_1 \in \{0,1\}^\lambda$, the probability that $G(x_0)$ and $G(x_1)$ are a collision on $h$ is $2^{-3\lambda}$. By taking a union bound over all such pairs $x_0, x_1$, the probability of there being any collision is at most $2^{-\lambda}$.

For the lossy mode of the function, we're given some random target that we denote by $x_0^*$. We choose an additional random input $x_1^*$ and "program" the hash function $h$ so that the values $G(x_0^*)$ and $G(x_1^*)$ collide, which ensures that $F_{\mathsf{fk}}(x_0^*) = F_{\mathsf{fk}}(x_1^*)$. Since $x_0^*$ and $x_1^*$ are treated symmetrically, the values $\mathsf{fk}, y = F_{\mathsf{fk}}(x_0^*) = F_{\mathsf{fk}}(x_1^*)$ do not disambiguate between them and hence preserve $\ell = 1$ bit of entropy in the target.

We can ensure that programming $h$ with a collision in lossy mode is computationally indistinguishable from choosing a random $h$ in injective mode, even given the target $x_0^*$. For concreteness, we consider the specific universal hash function $h_a(x) = \mathsf{chop}(a \cdot x)$, which performs a field multiplication over $\mathbb{F}_{2^{3\lambda+1}}$ and chops off the least significant bit. Using the standard representation of field elements, this implies that for all $y$ we have $\mathsf{chop}(y) = \mathsf{chop}(y+1)$. In that case, programming $h$ to ensure $G(x_0^*)$ collides with $G(x_1^*)$ means choosing $a = (G(x_0^*) - G(x_1^*))^{-1}$ so that $a \cdot G(x_0^*) = a \cdot G(x_1^*) + 1$ and therefore $h_a(G(x_0^*)) = h_a(G(x_1^*))$. But even if we're given the target $x_0^*$ the value $a = (G(x_0^*) - G(x_1^*))^{-1}$ is indistinguishable from uniform by the pseudorandomness of $G(x_1^*)$. Therefore the lossy mode of choosing $a$ is indistinguishable from the injective mode where $a$ is chosen uniformly at random. The above summarizes the entire construction and proof of security, highlighting its simplicity!

If we don't have an injective PRG, the same construction above already achieves a relaxed form of injectivity. Namely, the injective mode of the function uniquely determines the property $P(x) = G(x)$, while the lossy mode of the function loses 1-bit of information about the same property $P(x^*) = G(x^*)$ for the target $x^*$.

We observe that we can amplify the lossiness parameter $\ell$ arbitrarily via parallel repetition. Given a TLF $F_{\mathsf{fk}}$ with 1 bit of lossiness, we define $F'_{\mathsf{fk}'}(x_1, \ldots, x_\ell) = F_{\mathsf{fk}_1}(x_1)||\cdots||F_{\mathsf{fk}_\ell}(x_\ell)$ for $\mathsf{fk}' = (\mathsf{fk}_1, \ldots, \mathsf{fk}_\ell)$ to get $\ell$ bits of lossiness. While the lossiness amount can be made arbitrarily large, the lossiness rate (defined as the ratio of the lossiness $\ell$ to the input size) is stuck at $\frac{1}{\lambda}$ and is not improved by parallel repetition.

*Targeted All-Injective-But-One Functions (T-A**I**BOs).* We can also easily extend the basic construction to get a T-A**I**BO. For branches $\mathsf{tag} \in \{0,1\}^t$, we need to define a family of functions $F_{\mathsf{fk},\mathsf{tag}}(\cdot)$ such that, for a special branch $\mathsf{tag}^*$, the function $F_{\mathsf{fk},\mathsf{tag}^*}$ is lossy and for all other branches it is injective. We can achieve this generically from any TLF without branches where the injective function key $\mathsf{fk}$ is uniformly random, as is the case in our basic construction. We simply set $\mathsf{fk} = h$ to be a pairwise-independent hash function and then apply it to the value $\mathsf{tag}$ to derive a function key $\hat{\mathsf{fk}} = h(\mathsf{tag})$ for the basic TLF; the output of $F_{\mathsf{fk},\mathsf{tag}}(x)$ is then set to $F_{\hat{\mathsf{fk}}}(x)$. We program the hash so that the special branch $\mathsf{tag}^*$ maps to a lossy function key $\hat{\mathsf{fk}}^*$ for the target value $x^*$. The output of the hash on any other $\mathsf{tag} \neq \mathsf{tag}^*$ is random and independent, and therefore the resulting TLF function key $\hat{\mathsf{fk}}$ is injective with overwhelming probability.

*Targeted All-Lossy-But-One Functions (T-A**L**BOs).* Getting T-A**L**BOs is more involved. Recall that we need a family of functions $F_{\mathsf{fk},\mathsf{tag}}(\cdot)$ such that there is a special branch $\mathsf{tag}^*$ on which the function is injective and, on all other branches $\mathsf{tag} \neq \mathsf{tag}^*$, it is cumulatively targeted-lossy for some target $x^*$, meaning that the entire collection of outputs on *all* the lossy tags $\{F_{\mathsf{fk},\mathsf{tag}}(x^*) \ : \ \mathsf{tag} \neq \mathsf{tag}^*\}$ must lose $\ell$-bits of information about $x^*$. We start with an approach that was originally proposed by [BHK11], and later abstracted more explicitly in [DVW20], as a way of converting lossy (trapdoor) functions into all-lossy-but-one (trapdoor) functions in the non-targeted setting. We first describe this approach and then show how to adapt it to the targeted setting.

The basic idea of [BHK11, DVW20] is to rely on function composition. As a first step, assume we have a lossy (trapdoor) function $F_{\mathsf{fk}}$ where both the domain and the range are $\{0,1\}^n$, and in particular are the same. We can use it to construct an all-lossy-but-one (trapdoor) function $\overline{F}_{\mathsf{fk},\mathsf{tag}}$ with tags in $\{0,1\}^t$. We define the function key $\mathsf{fk} = ((\mathsf{fk}_1^0, \mathsf{fk}_1^1), \ldots, (\mathsf{fk}_t^0, \mathsf{fk}_t^1))$ to consist of $2t$ function keys for the underlying lossy function and we define

$$\overline{F}_{\mathsf{fk},\mathsf{tag}}(x) = (F_{\mathsf{fk}_t^{\mathsf{tag}_t}} \circ F_{\mathsf{fk}_{t-1}^{\mathsf{tag}_{t-1}}} \circ \cdots \circ F_{\mathsf{fk}_1^{\mathsf{tag}_1}})(x)$$

where $\mathsf{tag}_i$ denotes the $i$'th bit of $\mathsf{tag}$. We set the $t$ function keys $\mathsf{fk}_i^{\mathsf{tag}^*_i}$ corresponding to the injective branch $\mathsf{tag}^*$ to be injective and the other $t$ function keys $\mathsf{fk}_i^{1-\mathsf{tag}^*_i}$ to be lossy. Since the composition of injective functions is injective, it holds that $F_{\mathsf{fk},\mathsf{tag}^*}$ is an injective function. On the other hand, for any lossy branch $\mathsf{tag} \neq \mathsf{tag}^*$, there exists some $i$ such that $\mathsf{tag}_i \neq \mathsf{tag}^*_i$ and therefore one of the functions $F_{\mathsf{fk}_i^{\mathsf{tag}_i}}$ applied during the computation of $F_{\mathsf{fk},\mathsf{tag}}(x)$ will be lossy

and lose $\ell$ bits of information about its input, which is the same as losing $\ell$ bits of information about $x$ since its input is a permutation of $x$. This shows that for each lossy branch $\mathsf{tag} \neq \mathsf{tag}^*$, the function $F_{\mathsf{fk},\mathsf{tag}}$ is individually lossy. But we can even show that the lossy branches are cumulatively lossy. This is because the only information revealed about $x$ by all the $2^t - 1$ lossy branches cumulatively, $\{F_{\mathsf{fk},\mathsf{tag}}(x) \ : \ \mathsf{tag} \neq \mathsf{tag}^*\}$, can be deduced from the $t$ values one gets by applying the first $i-1$ injective functions followed by a lossy one in position $i$, for $i = 1, \ldots, t$. Each such output reveals at most $n - \ell$ bits of information about $x$ and hence in they reveal at most $t(n-\ell)$ bits in total. This gives lossiness $\ell' = n - t(n-\ell)$, which can be large if $\ell$ is very close to $n$ and $t$ is small relative to $n$; e.g, if $\ell = n(1 - o(1))$ and $t = o(n)$ then $\ell' = (1 - o(1))n$. Indeed, one can get such parameters from DDH.

Unfortunately, there are several issues with applying the above approach in our case. Firstly, our basic TLF does not have the same domain and range: it maps an input in $\{0,1\}^\lambda$ to an output in $\{0,1\}^{3\lambda}$. This makes it difficult to even syntactically rely on the above approach. Fortunately, this is relatively easy to fix. We can redefine our basic TLF with modified parameters $F_{\mathsf{fk}} \ : \ \{0,1\}^{3\lambda} \to \{0,1\}^{3\lambda}$ via $F_{\mathsf{fk}}(x) = h(G(x))$ where now we have $x \in \{0,1\}^{3\lambda}$, the injective PRG is of the form $G \ : \ \{0,1\}^{3\lambda} \to \{0,1\}^{3\lambda+1}$, and the unviersal hash functions are of the form $h \ : \ \{0,1\}^{3\lambda+1} \to \{0,1\}^{3\lambda}$. This lets us syntactically use $F_{\mathsf{fk}}$ in the above construction to define a family with branches. Unfortunately, now $F_{\mathsf{fk}}$ is no longer injective when $\mathsf{fk}$ is chosen in "injective mode". However, we can regain injectivity by first pre-processing a smaller input $x \in \{0,1\}^\lambda$ via an injective PRG $G' \ : \ \{0,1\}^\lambda \to \{0,1\}^{3\lambda}$. We define the overall function with branches $\overline{F}_{\mathsf{fk},\mathsf{tag}} \ : \ \{0,1\}^\lambda \to \{0,1\}^{3\lambda}$ via:

$$\overline{F}_{\mathsf{fk},\mathsf{tag}}(x) = (F_{\mathsf{fk}_t^{\mathsf{tag}_t}} \circ F_{\mathsf{fk}_{t-1}^{\mathsf{tag}_{t-1}}} \circ \cdots \circ F_{\mathsf{fk}_1^{\mathsf{tag}_1}})(G'(x)).$$

This preserves injectivity on the branch $\mathsf{tag}^*$ since, with overwhelming probability, each function component $F_{\mathsf{fk}_i^{\mathsf{tag}_i^*}}$ is injective over the domain of inputs $(F_{\mathsf{fk}_i^{\mathsf{tag}_{i-1}^*}} \circ \cdots \circ F_{\mathsf{fk}_1^{\mathsf{tag}_1^*}})(G'(x))$ of size $2^\lambda$. For targeted lossiness, we can now chose each of the targeted lossy keys $\mathsf{fk}_i^{1-\mathsf{tag}_i^*}$ with the target $x^*[i] = (F_{\mathsf{fk}_i^{\mathsf{tag}_{i-1}^*}} \circ \cdots \circ F_{\mathsf{fk}_1^{\mathsf{tag}_1^*}})(G'(x^*))$. This is enough to show that each lossy branch is individually targeted-lossy. Unfortunately, we don't get cumulative lossiness. Recall that the argument we employed in the previous paragraph only gave cumulative lossiness $\ell' = n - t(n-\ell)$, which was only meaningful when the initial lossiness $\ell$ was a large fraction of the domain size $n$. But in our case $\ell = 1$ and hence the above does not give us any meaningful bound on $\ell'$, even for tag size $t = 2$.

To solve the above issue, we need to control the lossiness more carefully to ensure that the leakages from different lossy tags don't add up. We do so by going under the hood of our basic TLF construction. We set $x_0^* = x^*$ to be the target and choose a uniformly random and independent $x_1^*$. We then choose all the lossy function keys $\mathsf{fk}_i^{1-\mathsf{tag}_i^*}$ to ensure that the two values $x_0^*, x_1^*$ collide on every lossy branch. We can do so by programming the universal hash function

in the $i$th lossy key to ensure that it has a collision on $G(x_0^*[i]), G(x_1^*[i])$ where $x_b^*[i] = (F_{\mathsf{fk}_i^{\mathsf{tag}_{i-1}^*}} \circ \cdots \circ F_{\mathsf{fk}_1^{\mathsf{tag}_1^*}})(G'(x_b^*))$. This guarantees that $F_{\mathsf{fk}_i^{1-\mathsf{tag}_i^*}}(x_0^*[i]) = F_{\mathsf{fk}_i^{1-\mathsf{tag}_i^*}}(x_1^*[i])$ and so $x_0^*$ and $x_1^*$ collide on every lossy branch. While this ensures cumulative lossiness, we now lose indistinguishability. The reason is that the randomness of $G(x_1^*[i])$ is used twice: once to define the lossy key $\mathsf{fk}_i^{1-\mathsf{tag}_i^*}$, and once to define $x_1^*[i+1]$, which is used to define the lossy key $\mathsf{fk}_{i+1}^{1-\mathsf{tag}_{i+1}^*}$. Since we're reusing the same randomness, the lossy keys for different values $i$ will appear correlated and can then be distinguished from injective keys. We can fix this issue by using two different PRGs $G_0$ and $G_1$ for the 0 functions $F_{\mathsf{fk}_i^0}$ and the 1 functions $F_{\mathsf{fk}_i^1}$ respectively. We need $G_0, G_1$ to each be injective and also to be mutually pseudorandom so that, for a random $x$ the values $G_0(x), G_1(x)$ look like random and independent values.[4] With this modification, we preserve indistinguishability. This is because, the lossy function key $\mathsf{fk}_i^{1-\mathsf{tag}_i^*}$ now only relies on $G^{1-\mathsf{tag}^*}(x_1^*[i])$ while the value $x_1^*[i+1]$ used to define $\mathsf{fk}_{i+1}^{1-\mathsf{tag}_{i+1}^*}$ only relies on $G^{\mathsf{tag}^*}(x_1^*[i])$, and hence we can argue that the two values look random and independent.

*T-A**L**BO with Relaxed Injectivity.* We can also get a T-A**L**BO with relaxed injectivity by using the same construction as above with standard PRGs rather than injective PRGs, and therefore from one-way functions. The observation is that, when we program the lossy mode to ensure that the target $x^* = x_0^*$ collides with some random $x_1^*$ on every lossy tag, it's very unlikely that $x_0^*$ and $x_1^*$ would collide on the injective tag, even when the PRG is not injective. Therefore, although the injective output $F_{\mathsf{fk},\mathsf{tag}^*}(x^*)$ may not uniquely determine $x^*$, we ensure that $F_{\mathsf{fk},\mathsf{tag}^*}(x^*)$ has 1-bit of entropy even given $\{F_{\mathsf{fk},\mathsf{tag}}(x^*) : \mathsf{tag} \neq \mathsf{tag}^*\}$. In other words, the injective mode of the function reveals at least 1 bit of information about $x^*$ that was lost by all the lossy evaluations on $x^*$. In fact, we can even ensure that the above holds if we shorten the output size of the function to just 1 bit (in which case, the function certainly can't be injective). We do so by applying a universal hash function with 1-bit output at the end, and programming it to ensure that $F_{\mathsf{fk},\mathsf{tag}^*}(x_0^*)$ and $F_{\mathsf{fk},\mathsf{tag}^*}(x_1^*)$ hash to different bits. This ensures 1 bit of entropy in a 1 bit output, and therefore the output of the injective branch is *uniformly random*, even given the outputs of all the lossy branches. We can amplify from 1 bit to many bits via parallel repetition.

*Applications of T-A**L**BOs.* We notice that T-A**L**BOs with relaxed injectivity directly give us *pseudo-entropy functions*, just by relabeling the components. We define the secret key $k$ of the pseudo-entropy function as $k = (\mathsf{fk}, s)$ to consist of a function key $\mathsf{fk}$ for a T-A**L**BO and a uniformly random input $s$ for it. We then define the pseudo-entropy function $f_k(x) = F_{\mathsf{fk},\mathsf{tag}=x}(s)$ which interprets its input $x$ as a branch and evaluates the T-A**L**BO on $s$. For any input $x^*$ chosen

[4] Equivalently, we can think of $G_0(x), G_1(x)$ as the left/right halves of a single PRG $G(x)$.

a-priori, we can choose $k = (\mathsf{fk}, s)$ by selecting a random $s$ and choosing $\mathsf{fk}$ with the injective branch $x^*$ and the target $s$. This guarantees the properties of a pseudo-entropy function: the value $k$ chosen this way is indistinguishable from an honestly chosen $k$ that is independent of $x^*$, but ensures that $f_k(x^*)$ has $\ell$ bits of statistical entropy even given $f_k(x)$ for all $x \neq x^*$. This shows that T-A**L**BOs directly give pseudo-entropy functions. In fact, this gives a pseudo-entropy function where the key $k$ consists of a uniformly random secret component $s$ and a public but carefully chosen component $\mathsf{fk}$ defined in terms of $s$ and the point $x^*$ on which we want to ensure statistical entropy. Conversely, a pseudo-entropy function of this form also gives a T-A**L**BO. By using a T-A**L**BO where lossiness $\ell$ is equal to the output size (as we showed can be done above), we can even ensure that $f_k(x^*)$ is uniformly random given $f_k(x)$ for all $x \neq x^*$.

Our applications to *leakage-resilient MACs, leakage-resilient symmetric-key encryption, and extractors for extractor-dependent sources* follow as interesting applications of pseudo-entropy functions.

For *selective-opening security*, we notice that our pseudo-entropy function has an additional feature. Not only can we ensure that $f_k(x^*)$ is uniformly random given $f_k(x)$ for all $x \neq x^*$, but for any output $y$ we can even efficiently find a key $k_y$ such that $f_{k_y}(x^*) = y$ and $f_{k_y}(x) = f_k(x)$ for all $x \neq x^*$. Intuitively, this additional feature gives us the ability to "equivocate", which is used to get selective-opening security. In particular, a simulator can find a key $k_y$ to open a challenge ciphertext to any value it wants, and looks consistent even to an adversary that got access to a CPA oracle.

*Application of T-A**I**BOs to CCA Security.* We also give an application of T-A**I**BOs to CCA-secure encryption from any trapdoor function with a sufficient high level of security. We describe a simplified version of this result, and the main body gives a more general treatment. Let $F_{\mathsf{fk},\mathsf{tag}}$ be a T-A**I**BO with $\lambda$-bit input and $\ell = 1$ bits of lossiness, as we constructed from injective pseudorandom generators. Let $f_{\mathsf{pk}}$ be a family of trapdoor functions (not necessarily permutations) with input length $n = \lambda^3$. Our CCA encryption public key consists of the pair $(\mathsf{pk}, \mathsf{fk})$ and the secret key is the trapdoor of the trapdoor function. The encryption procedure selects a random $r \in \{0,1\}^n$ and parses it as $r = (r_1, \ldots, r_d)$ with $d = \lambda^2$ and $r_i \in \{0,1\}^\lambda$. It also selects a one-time signature key pair $(\mathsf{vk}, \mathsf{sk})$. It compute $y = f_{\mathsf{pk}}(r)$ and $y_1 = F_{\mathsf{fk},\mathsf{vk}}(r_1), \ldots, y_{\lambda^2} = F_{\mathsf{fk},\mathsf{vk}}(r_d)$ then uses a Goldreich-Levin hardcore bit of $r$ to one-time pad the message and signs everything under $\mathsf{vk}$. The decryption procedure checks the signature, inverts $y$ to recover $r$ and checks that $y_1, \ldots, y_d$ were computed correctly: if so it recovers the hardcore bit and decrypts the message, else it rejects.

To prove CCA security, we select $\mathsf{fk}$ to be lossy on the branch $\mathsf{tag} = \mathsf{vk}$ and the target value $r$ that correspond to the challenge ciphertext. We can then simulate the decryption procedure without knowing the trapdoor $\mathsf{td}$ by brute-force inverting all the the values $y_i = F_{\mathsf{fk},\mathsf{vk}}(r_i)$ in $\tilde{O}(2^\lambda)$ time. In the challenge ciphertext, the value $r = (r_1, \ldots, r_d)$ has $d = \lambda^2$ bits of entropy even given $y_1, \ldots, y_d$. We argue that this makes it hard to recover $r$ even given the trapdoor function output $f_{\mathsf{pk}}(r)$ and the ability to run in $\tilde{O}(2^\lambda)$ time. We show

that this follows from very strong exponential hardness of the trapdoor function: we need to assume that for input length $n = \lambda^3$ no adversary running in time $\tilde{O}(2^\lambda)$ can invert the function with better than $\frac{2^{\lambda^2}}{2^{\lambda^3}}$ probability. While this is a strong assumption, note that the trivial attack that tries $2^\lambda$ random inputs only has success probability $\frac{2^\lambda}{2^{\lambda^3}}$ and generic non-uniform attacks [DGK17] can't do better than $\frac{2^{\widetilde{O}(\lambda)}}{2^{\lambda^3}}$.

### 1.3 Relation to Distributed Point Functions

We observe an interesting connection between T-A**L**BOs (with relaxed injectivity), pseudo-entropy functions, and distributed-point functions (DPFs) [GI14, BGI15]. In fact, even though the notions look very different and were introduced with different goals in mind, they are essentially equivalent. We already discussed the connection between T-A**L**BOs and pseudo-entropy functions, and so we now show the connection to DPFs.

Distributed point functions were defined in the context of 2-server private information retrieval (PIR). They consist of a function family $f_k : [N] \to \{0,1\}$. Given some target $x^* \in [N]$, it should be possible to choose two keys $k_0, k_1$ such that $f_{k_0}(x^*) \neq f_{k_1}(x^*)$ differ on the target point, but for all other points $x \neq x^*$ they are the same $f_{k_0}(x^*) = f_{k_1}(x^*)$. Each of the keys $k_0, k_1$ should individually computationally hide the value $x^*$. This gives 2-server PIR. When a client wants to retrieve a value $\mathsf{DB}[x^*]$ at the location $x^* \in [N]$ of a database $\mathsf{DB} \in \{0,1\}^N$, it chooses the two keys $k_0, k_1$ using a target $x^*$ and sends $k_b$ to server $b$. Each server $b$ computes $y_b = \bigoplus_{x \in [N]} f_{k_b}(x) \cdot \mathsf{DB}[x]$ and the client computes $y_0 \oplus y_1 = \mathsf{DB}[x^*]$. Neither server individually learns anything about the location $x^*$ by the hiding property of the DPF.

A pseudo-entropy function with 1-bit output and 1-bit entropy almost already gives a DPF. If we select the key $k$ to preserve entropy on $x^*$, then $f_k(x^*)$ has 1 bit of entropy even given $f_k(x)$ for all $x \neq x^*$. That means that there must be some key $k'$ such that $f_{k'}(x^*) \neq f_k(x^*)$ but $f_{k'}(x) = f_k(x)$ for all $x \neq x^*$. We can define $k_0 = k$ and $k_1 = k'$ to get the two DPF keys for the point $x^*$. The only difficulty is ensuring that we can kind $k'$ efficiently. Recall that in our construction of pseudo-entropy functions for T-A**L**BOs, we set $k = (\mathsf{fk}, s)$ where $\mathsf{fk}$ is a function key of a T-A**L**BO with the "injective" branch $x^*$ and the target input $s$. When we choose $\mathsf{fk}$, our T-A**L**BO construction in turn sets $s_0 = s$, picks a random $s_1$ and ensures that the function outputs collide on $s_0, s_1$ for all branches $x \neq x^*$ but differ on the branch $x^*$. Therefore, we can efficiently set $k_0 = (\mathsf{fk}, s_0)$ and $k_1 = (\mathsf{fk}, s_1)$.

Interestingly, although our construction of T-A**L**BOs was initially inspired by the works of [BHK11, DVW20], we observe in retrospect that it is very similar to the construction of DPFs in [BGI15]. Indeed the function composition construction of T-A**L**BOs using two PRGs $G_0, G_1$ is similar to the GGM construction of PRFs from PRGs [GGM86], and the use of hash functions $h$ is similar to the use of "correction words" in the adaptation of GGM to DPFs in [BGI15].

We hope that the connections between all these notions helps bring about a better understanding of each of them. The fact that completely different motivations and construction ideas surreptitiously converged to yield related notions and constructions should perhaps be viewed as a good indication of just how fundamental these notions are.

## 2 Preliminaries

*Basic Notation.* For an integer $N$, we let $[N] \coloneqq \{1, 2, \ldots, N\}$. For a set $S$ we let $x \leftarrow S$ denote sampling $x$ uniformly at random from $S$. For a distribution $\mathcal{S}$ we let $x \leftarrow \mathcal{S}$ denote sampling $x$ according to the distribution. We will denote the security parameter by $\lambda$. We say a function $f(\lambda)$ is negligible, denoted $f(\lambda) = \mathrm{negl}(\lambda)$, if $f(\lambda) = O(\lambda^{-c})$ for every constant $c > 0$. A function is $f(\lambda)$ is polynomial, denoted $f(\lambda) = \mathrm{poly}(\lambda)$, if $f(\lambda) = O(\lambda^{c})$ for some constant $c > 0$. For a randomized algorithm $\mathsf{A}$, we will sometimes explicitly denote the randomness $\mathsf{coins}$ it uses, writing $\mathsf{A}(x; \mathsf{coins})$. We will write $D_1 \overset{\mathrm{c}}{\approx} D_2$ if the (ensembles of) distrubtions $D_1$ and $D_2$ are computationally indistinguishable.

*Information Theory.* For two random variables $X, Y$ with support $\mathrm{supp}(X)$ and $\mathrm{supp}(Y)$ respectively, we define their statistical distance $\mathsf{SD}(X, Y)$ as

$$\mathsf{SD}(X, Y) \coloneqq \sum_{u \in \mathrm{supp}(X) \cup \mathrm{supp}(Y)} \frac{1}{2} |\Pr[X = u] - \Pr[Y = u]|.$$

Two ensembles of random variables $X = \{X_\lambda\}_\lambda, Y = \{Y_\lambda\}_\lambda$ are statistically close if $\mathsf{SD}(X_\lambda, Y_\lambda) = \mathrm{negl}(\lambda)$.

The min-entropy $H_\infty(X)$ of a random variable $X$ is defined as

$$H_\infty(X) \coloneqq -\log(\max_{x \in \mathrm{supp}(X)} Pr[X = x]).$$

Following Dodis et al. [DORS08], we define the (average) conditional min-entropy of $X$ given $Y$ as: $H_\infty(X|Y) = -\log\left(\mathbb{E}_{y \leftarrow Y}\left[2^{-H_\infty(X|Y=y)}\right]\right)$. Note that $H_\infty(X|Y) = k$ iff the optimal strategy for guessing $X$ given $Y$ succeeds with probability $2^{-k}$.

## 3 Definitions

### 3.1 Targeted Lossy Functions (TLFs)

We first define our basic notion of targeted lossy functions (TLF).

**Definition 1 (Targeted Lossy Functions).** *A targeted lossy function (TLF) function family with input length $n = n(\lambda)$, output length $m \geq n$ and lossiness parameter $\ell = \ell(\lambda)$ consists of PPT algorithms* $(\mathsf{InjectiveGen}, \mathsf{LossyGen}, F)$ *with the following syntax:*

- $\mathsf{fk} \leftarrow \mathsf{InjectiveGen}(1^\lambda)$*: generates a function key* $\mathsf{fk}$*.*
- $\mathsf{fk} \leftarrow \mathsf{LossyGen}(1^\lambda, x^*)$*: on input a target value* $x^* \in \{0,1\}^n$*, generates a function key* $\mathsf{fk}$*.*
- $y = F_{\mathsf{fk}}(x)$*: a deterministic algorithm which, on input* $\mathsf{fk}$ *along with a value* $x \in \{0,1\}^n$*, outputs* $y \in \{0,1\}^m$*.*

*We require the following properties:*

**Injectivity:** *With overwhelming probability over the choice of* $\mathsf{fk} \leftarrow \mathsf{InjectiveGen}(1^\lambda)$*, the function* $F_{\mathsf{fk}}$ *is injective over its domain* $\{0,1\}^n$*.*

**$\ell$-Lossiness:** *For random variables* $x^* \leftarrow \{0,1\}^n$*,* $\mathsf{fk} \leftarrow \mathsf{LossyGen}(1^\lambda, x^*)$*, we have*

$$H_\infty(x^* \mid \mathsf{fk}, F_{\mathsf{fk}}(x^*)) \geq \ell.$$

**Indistinguishability:** *For all* $x^* \in \{0,1\}^n$*, we have the computational indistinguishability*

$$(x^*, \mathsf{fk}_{inj}) \stackrel{\mathrm{c}}{\approx} (x^*, \mathsf{fk}_{loss})$$

*where* $\mathsf{fk}_{inj} \leftarrow \mathsf{InjectiveGen}(1^\lambda)$ *and* $\mathsf{fk}_{loss} \leftarrow \mathsf{LossyGen}(1^\lambda, x^*)$*.*

*Relaxing Injectivity.* We can also define a variant of TLFs with *relaxed injectivity*, where we require the injective mode to only uniquely determine some property $P(x)$ while lossy mode loses $\ell$-bits of information on $P(x^*)$ for the target $x^*$. In particular, we require that there exists some function $P : \{0,1\}^* \to \{0,1\}^*$ for which the following holds:

- *Relaxed Injectivity:* With overwhelming probability over the choice of $\mathsf{fk} \leftarrow \mathsf{InjectiveGen}(1^\lambda)$ it holds that for all $x, x' \in \{0,1\}^n$ if $F_{\mathsf{fk}}(x) = F_{\mathsf{fk}}(x')$ then $P(x) = P(x')$.
- *$\ell$-Lossiness:* For random variables $x^* \leftarrow \{0,1\}^n$, $\mathsf{fk} \leftarrow \mathsf{LossyGen}(1^\lambda, x^*)$, we have

$$H_\infty(P(x^*) \mid \mathsf{fk}, F_{\mathsf{fk}}(x^*)) \geq \ell.$$

### 3.2 Targeted All-Lossy-But-One Functions (T-ALBO)

Next, we define a tagged version of TLFs, that we name targeted all-lossy-but-one functions (T-A**L**BO).

**Definition 2 (T-ALBO).** *A targeted all-lossy-but-one (T-A**L**BO) function family with input length* $n = n(\lambda)$*, output length* $m \geq n$*, tag length* $t = t(\lambda)$ *and lossiness parameter* $\ell = \ell(\lambda)$*, consists of PPT algorithms* $(\mathsf{InjectiveGen}, \mathsf{LossyGen}, F)$ *with the following syntax:*

- $\mathsf{fk} \leftarrow \mathsf{InjectiveGen}(1^\lambda)$*: generates a function key* $\mathsf{fk}$*.*
- $\mathsf{fk} \leftarrow \mathsf{LossyGen}(1^\lambda, \mathsf{tag}^*, x^*)$*: on input* $\mathsf{tag}^* \in \{0,1\}^t, x^* \in \{0,1\}^n$*, generates a function key* $\mathsf{fk}$*.*
- $y = F_{\mathsf{fk},\mathsf{tag}}(x)$*: a deterministic algorithm, which, on input* $\mathsf{fk}, \mathsf{tag}$ *along with a value* $x \in \{0,1\}^n$*, outputs* $y \in \{0,1\}^m$*.*

*We require the following properties:*

**Injectivity:** *With overwhelming probability over the choice of* $\mathsf{fk} \leftarrow \mathsf{InjectiveGen}(1^\lambda)$*, for all* $\mathsf{tag} \in \{0,1\}^t$*, the function* $F_{\mathsf{fk},\mathsf{tag}}$ *is injective over the domain* $\{0,1\}^n$*. Moreover, for any* $\mathsf{tag}^*, x^*$*, with overwhelming probability over the choice of* $\mathsf{fk} \leftarrow \mathsf{LossyGen}(1^\lambda, \mathsf{tag}^*, x^*)$*, the function* $F_{\mathsf{fk},\mathsf{tag}^*}$ *is injective.*

**$\ell$-Lossiness:** *For all* $\mathsf{tag}^* \in \{0,1\}^t$ *and random* $x^* \leftarrow \{0,1\}^n$*,* $\mathsf{fk} \leftarrow \mathsf{LossyGen}(1^\lambda, \mathsf{tag}^*, x^*)$*, we have* $H_\infty(x^* \mid \mathsf{fk}, \{F_{\mathsf{fk},\mathsf{tag}}(x^*) \ : \ \mathsf{tag} \neq \mathsf{tag}^*\}) \geq \ell$*.*

**Indistinguishability:** *For all* $\mathsf{tag}^* \in \{0,1\}^t$ *and all* $x^* \in \{0,1\}^n$*, we have*

$$(\mathsf{tag}^*, x^*, \mathsf{fk}_{inj}) \stackrel{\text{c}}{\approx} (\mathsf{tag}^*, x^*, \mathsf{fk}_{loss})$$

*where* $\mathsf{fk}_{inj} \leftarrow \mathsf{InjectiveGen}(1^\lambda)$ *and* $\mathsf{fk}_{loss} \leftarrow \mathsf{LossyGen}(1^\lambda, \mathsf{tag}^*, x^*)$*.*

*Relaxing Injectivity: Entropy-Preserving T-A**L**BOs.* We can also relax injectivity in much the same way as we did for TLFs, by requiring that injective mode uniqely determines some property $P(x)$ while lossy mode loses information on $P(x^*)$. Here, we can even allow the property $P$ to depend on $\mathsf{fk}, \mathsf{tag}^*$. In this case, without loss of generality, we can set $P(x) = F_{\mathsf{fk},\mathsf{tag}^*}(x)$ to be the output on the "injective" tag, and therefore it tautologically holds that $F_{\mathsf{fk},\mathsf{tag}^*}(x)$ determines $P(x)$. Hence this notion just requires that seeing the output of the function on input $x^*$ over all lossy branches $\mathsf{tag} \neq \mathsf{tag}^*$ preserves some entropy of the output $F_{\mathsf{fk},\mathsf{tag}^*}(x^*)$ on the "injective" branch $\mathsf{tag}^*$. We call this notion *entropy preserving*. This notion also meaningfully allows us to make the output much smaller than the input size, and potentially just 1-bit.

**Definition 3 (Entropy-Preserving T-ALBO.).** *An* entropy-preserving *T-A**L**BO with input length* $n = n(\lambda)$*, output length* $m = m(\lambda)$*, tag length* $t = t(\lambda)$ *and lossiness parameter* $\ell = \ell(\lambda)$*, consists of algorithms* $(\mathsf{InjectiveGen}, \mathsf{LossyGen}, F)$*, satisfying indistinguishability and the following entropy-preserving property. For any fixed* $\mathsf{tag}^* \in \{0,1\}^t$ *we have:*

$$H_\infty(F_{\mathsf{fk}}(\mathsf{tag}^*, x^*) \mid \mathsf{fk}, \{F_{\mathsf{fk},\mathsf{tag}}(x^*) \ : \ \mathsf{tag} \neq \mathsf{tag}^*\}) \geq \ell,$$

*where* $x^* \leftarrow \{0,1\}^n$*,* $\mathsf{fk} \leftarrow \mathsf{LossyGen}(1^\lambda, \mathsf{tag}^*, x^*)$*.*

*We say that* $(\mathsf{InjectiveGen}, \mathsf{LossyGen}, F)$ *is* maximally entropy-preserving *if* $\ell = m$*, where* $m$ *is the output size of the T-A**L**BO.*

### 3.3 Targeted All-Injective-But-One Functions (T-AIBO)

Last, we define another tagged variant of TLFs that we call *targeted all-injective-but-one lossy functions* (T-A**I**BO). In a T-A**I**BO, the branches $\mathsf{tag} \neq \mathsf{tag}^*$ are injective, whereas only $\mathsf{tag}^*$ corresponds to a lossy branch.

**Definition 4 (T-AIBO).** *A targeted all-injective-but-one T-A**I**BO function family with input length* $n = n(\lambda)$*, output length* $m \geq n$*, tag length* $t = t(\lambda)$ *and lossiness parameter* $\ell = \ell(\lambda)$*, consists of PPT algorithms* $(\mathsf{InjectiveGen}, \mathsf{LossyGen}, F)$ *with the following syntax:*

- $\mathsf{fk} \leftarrow \mathsf{InjectiveGen}(1^\lambda)$*: generates a function key* $\mathsf{fk}$*.*
- $\mathsf{fk} \leftarrow \mathsf{LossyGen}(1^\lambda, \mathsf{tag}^*, x^*)$*: on input* $\mathsf{tag}^* \in \{0,1\}^t, x^* \in \{0,1\}^n$*, generates a function key* $\mathsf{fk}$*.*
- $y = F_{\mathsf{fk},\mathsf{tag}}(x)$*: a deterministic polynomial time algorithm, which, on input* $\mathsf{fk}, \mathsf{tag}$ *along with a value* $x \in \{0,1\}^n$ *and outputs* $y \in \{0,1\}^m$*.*

*We require the following properties:*

**Injectivity on injective branches:** *With overwhelming probability over the choice of* $\mathsf{fk} \leftarrow \mathsf{InjectiveGen}(1^\lambda)$*, we have that for all tags* $\mathsf{tag} \in \{0,1\}^t$*, the function* $F_{\mathsf{fk},\mathsf{tag}}$ *is injective over the domain* $\{0,1\}^n$*. Moreover, for any* $\mathsf{tag}^*, x^*$*, with overwhelming probability over the choice of* $\mathsf{fk} \leftarrow \mathsf{LossyGen}(1^\lambda, \mathsf{tag}^*, x^*)$*, we have that for all tags* $\mathsf{tag} \neq \mathsf{tag}^*$*, the function* $F_{\mathsf{fk},\mathsf{tag}}$ *is injective.*

**$\ell$-Lossiness:** *For any* $\mathsf{tag}^* \in \{0,1\}^t$ *and random* $x^* \leftarrow \{0,1\}^n$*,* $\mathsf{fk} \leftarrow \mathsf{LossyGen}(1^\lambda, \mathsf{tag}^*, s^*)$*, we have* $H_\infty(x^* \mid \mathsf{fk}, F_{\mathsf{fk},\mathsf{tag}^*}(x^*)) \geq \ell$*.*

**Indistinguishability:** *For any* $\mathsf{tag}^* \in \{0,1\}^t$ *and* $x^* \in \{0,1\}^n$*, we have the computational indistinguishability*

$$(\mathsf{tag}^*, x^*, \mathsf{fk}_{inj}) \overset{\mathrm{c}}{\approx} (\mathsf{tag}^*, x^*, \mathsf{fk}_{loss})$$

*where* $x^* \leftarrow \{0,1\}^n, \mathsf{fk}_{inj} \leftarrow \mathsf{InjectiveGen}(1^\lambda)$ *and* $\mathsf{fk}_{loss} \leftarrow \mathsf{LossyGen}(1^\lambda, \mathsf{tag}^*, x^*)$*.*

We could also relax injectivity as we did for TLFs and T-A**L**BOs. Since we do not consider this notion in the paper, we omit it for simplicity.

## 4 Constructions

In this section we present our constructions of TLFs and its variants. In Section 4.1, we give the construction of basic TLFs (Theorem 1). Then, we show in Section 4.2 our construction of a T-A**I**BO (Theorem 2). Finally, in Section 4.3, we build both a T-A**L**BO (Theorem 3) and a maximally entropy-preserving T-A**L**BO (Theorem 4).

### 4.1 Targeted Lossy Functions

We start with our base construction of TLF. We prove the following:

**Theorem 1 (TLFs from Injective PRGs).** *Let* $\ell = \ell(\lambda)$ *be a polynomial. Assuming the existence of injective PRGs, there exists a TLF with input length* $n = \ell\lambda$*, output length* $m = 3\ell\lambda$ *and lossiness* $\ell$*.*

Let $G : \{0,1\}^\lambda \to \{0,1\}^{3\lambda+1}$ be a PRG. Let $\mathbb{F} = \mathbb{F}_{2^{3\lambda+1}}$ be the field of size $2^{3\lambda+1}$. We represent elements of $\mathbb{F}$ in the standard manner as $3\lambda+1$ coefficients of polynomials in $\mathbb{F}_2[X]/(f)$ for some appropriate polynomial $f$, so that adding elements in $\mathbb{F}$ can be performed by adding their coefficients component wise. For $a \in \mathbb{F}$, represented as a bit string of length $3\lambda+1$, define $\mathsf{chop}(a)$ as the first $3\lambda$ bits of the representation of $a$ (i.e., the last bit, corresponding to the constant

term of the polynomial, is chopped off). Let $e = 1_{\mathbb{F}} \in \mathbb{F}$ be the field element that has 0s in the first $3\lambda$ positions and 1 in the last position; we will keep the notation $e$ to clarify that it is a field element. Note that for all $x \in \mathbb{F}$ we have $\mathsf{chop}(x+e) = \mathsf{chop}(x)$, and hence for all $x_1, x_2$, $\mathsf{chop}(x_1) = \mathsf{chop}(x_2)$ if and only if $x_1 = x_2$ or $x_1 = x_2 + e$.

We first construct an LTF $(\mathsf{InjectiveGen}, \mathsf{LossyGen}, F)$ with input length $n = \lambda$ and output length $m = 3\lambda$, and lossiness 1 as follows.

- $\mathsf{fk} \leftarrow \mathsf{InjectiveGen}(1^\lambda)$: Sample $a \leftarrow \mathbb{F}$ and output $\mathsf{fk} = a$.
- $\mathsf{fk} \leftarrow \mathsf{LossyGen}(1^\lambda, x^*)$: On input $x^* \in \{0,1\}^\lambda$, set $x_0^* := x^*$ and sample $x_1^* \leftarrow \{0,1\}^\lambda \setminus \{x^*\}$. Set $a = e \cdot (G(x_0^*) - G(x_1^*))^{-1}$, and output $\mathsf{fk} = a$.
- $F_{\mathsf{fk}}(x) = \mathsf{chop}(a \cdot G(x)) \in \{0,1\}^{3\lambda}$.

**Claim 1.** *Suppose $G : \{0,1\}^\lambda \to \{0,1\}^{3\lambda+1}$ is an injective PRG. Then $(\mathsf{InjectiveGen}, \mathsf{LossyGen}, F)$ is a TLF with input length $n = \lambda$, output length $m = 3\lambda$ and lossiness $\ell = 1$.*

*Proof.* Note that in lossy mode, $a$ is well-defined by injectivity of $G$. We prove injectivity, $\ell$-lossiness and indistinguishability.

**Injectivity.** Fix any $x_0 \neq x_1 \in \mathbb{F}$. By injectivity of $G$, we have $G(x_0) \neq G(x_1)$. Therefore, $F_{\mathsf{fk}}(x_0) = F_{\mathsf{fk}}(x_1)$ iff $a \cdot G(x_0) = a \cdot G(x_1) + e$ or $a = 0$ iff $a = e \cdot (G(x_0) - G(x_1))^{-1}$ or $a = 0$ . This happens with probability $2/|\mathbb{F}|$ over the randomness of $\mathsf{fk} \leftarrow \mathsf{InjectiveGen}(1^\lambda)$. By taking a union bound over the pairs of distinct inputs $x_0, x_1$ (of which there are fewer than $2^{2\lambda}$), we obtain that the probability over $\mathsf{fk} \leftarrow \mathsf{InjectiveGen}(1^\lambda)$ that $F_{\mathsf{fk}}$ is not injective is at most $\frac{2^{2\lambda+1}}{|\mathbb{F}|} = \frac{1}{2^\lambda}$.

**$(\ell = 1)$-Lossiness.** Let $\mathsf{fk} \leftarrow \mathsf{LossyGen}(1^\lambda, x_0^*)$ for a random $x_0^* \leftarrow \{0,1\}^\lambda$, and denote by $x_1^* \leftarrow \{0,1\}^\lambda \setminus \{x_0^*\}$ the internal randomness used by $\mathsf{LossyGen}$. The view $(\mathsf{fk}, F_{\mathsf{fk}}(x^*))$ is a randomized function of the unordered set $\{x_0^*, x_1^*\}$, where $x_0^*, x_1^* \leftarrow \{0,1\}^\lambda$ conditioned on $x_0^* \neq x_1^*$. Namely, such a function picks $b \leftarrow \{0,1\}$ at random, computes $\mathsf{fk} \leftarrow \mathsf{LossyGen}(1^\lambda, x_b^*; x_{1-b}^*)$ and outputs $(\mathsf{fk}, F_{\mathsf{fk}}(x_b^*))$. In particular, a data processing inequality gives:

$$H_\infty(x_0^* \mid \mathsf{fk}, F_{\mathsf{fk}}(x_0^*)) \geq H_\infty(x_b^* \mid \{x_0^*, x_1^*\}) \geq 1,$$

where $x_0^*, x_1^* \leftarrow \{0,1\}^\lambda$ conditioned on $x_0^* \neq x_1^*$, and $b \leftarrow \{0,1\}$.

**Indistinguishability.** We define a hybrid experiment where $\mathsf{LossyGen}$ is modified as follows:

- $\widetilde{\mathsf{LossyGen}}(1^\lambda, x^*)$: Set $x_0^* := x^*$ and select $u_1^* \leftarrow \mathbb{F}$. If $u_1^* = G(x_0^*)$, output $\mathsf{fk} = 0$. Let $a = e \cdot (G(x_0^*) - u_1^*)^{-1}$.

The output of $\widetilde{\mathsf{LossyGen}}$ is indistinguishable from the output of $\mathsf{LossyGen}$ by PRG security of $G$, noting that $u_1^* = G(x_0^*)$ with negligible probability $1/2^{3\lambda+1}$ over the randomness of $u_1^*$ alone.

Moreover, even given $x^*$, the output of $\widetilde{\mathsf{LossyGen}}$ is uniformly random in $\mathbb{F}$ over the choice of $u_1^*$ alone, and is therefore identically distributed as the output of $\mathsf{InjectiveGen}$.

*Amplifying (absolute) lossiness.* The construction above only have lossiness 1. We note here that we can amplify this lossiness, which gives Theorem 1.

The idea is that (absolute) lossiness can be amplified by partitioning a (longer) input into blocks and applying an independent TLF on each chunk. Suppose TLF $(\mathsf{InjectiveGen}, \mathsf{LossyGen}, F)$ is a TLF with input size $n$, output size $m$ and lossiness $\ell$. Let $k = k(\lambda)$ be a polynomial. The modified scheme is defined as follows:

- $\mathsf{fk} \leftarrow \overline{\mathsf{InjectiveGen}}(1^\lambda)$: For all $i \in [k]$, compute $\mathsf{fk}_i \leftarrow \mathsf{InjectiveGen}$. Output $\{\mathsf{fk}_i\}_{i\in[k]}$.
- $\mathsf{fk} \leftarrow \overline{\mathsf{LossyGen}}(1^\lambda, x^*)$: On input $x^* \in \{0,1\}^{kn}$, parse $x^* = x_1 \| \cdots x_k \in \{0,1\}^{kn}$ where $x_i \in \{0,1\}^n$ for all $i \in [k]$. For all $i \in [k]$, compute $\mathsf{fk}_i \leftarrow \mathsf{LossyGen}(s_i)$. Output $\{\mathsf{fk}_i\}_{i\in[k]}$.
- $\overline{F}_{\mathsf{fk}}(x)$ : On input $x \in \{0,1\}^{kn}$, parse $x = x_1 \| \cdots x_k \in \{0,1\}^{kn}$ where $x_i \in \{0,1\}^n$ for all $i \in [k]$. For all $i \in [k]$, compute $y_i = F_{\mathsf{fk}_i}(x_i)$. Output $(y_1 \| \cdots y_k)$.

The resulting scheme is a TLF with input size $kn$, output size $km$ and lossiness $k\ell$. This is at the cost of making the input longer, and therefore doesn't affect the lossiness rate. Applying the above to our construction from Claim 1, we obtain Theorem 1.

*Remark: Relaxed Injectivity.* If we take our construction of TLFs above but remove the requirement that the PRG is injective, we get relaxed injectivity with the property $P(x) = G(x)$. The proofs are essentially identical.

### 4.2 T-AIBOs

We describe our construction of T-A**I**BO. We prove the following:

**Theorem 2 (T-AIBOs from Injective PRGs).** *Let $\ell = \ell(\lambda)$ and $t = t(\lambda)$ be polynomials. Assuming the existence of injective PRGs, there exists a T-A**I**BO with input length $n = \ell\lambda$, tag length $t$, output length $m = \ell \cdot (3\lambda + t)$ and lossiness $\ell$.*

We build on our construction of TLF from Section 4.1. Recall that we built our TLF as $F_{\mathsf{fk}}(s) = \mathsf{chop}(a \cdot G(s))$, where $a \in \mathbb{F}$ forms the key $\mathsf{fk}$. In order to build a T-A**I**BO, we now compute $a_{\mathsf{tag}} = h_k(\mathsf{tag})$ where $h$ is a pairwise independent hash function.

More formally, let $t = t(\lambda)$ be the tag length. Let $n = 3\lambda + t + 1$, and let $\mathbb{F} = \mathbb{F}_{2^n}$. We consider elements $\mathsf{tag} \in \{0,1\}^t$ as elements of $\mathbb{F}$ (for instance by considering the coefficient embedding of $\mathbb{F}$ as in Section 4.1), over which we define the pairwise independent hash function

$$h_{u,v}(\mathsf{tag}) = u \cdot \mathsf{tag} + v \in \mathbb{F},$$

where $u, v \in \mathbb{F}$. We will use the following useful algorithm related to $h$:

– $\mathsf{Equivocate}(\mathsf{tag}, y)$: on input $\mathsf{tag} \in \mathbb{F}$ and $y \in \mathbb{F}$, sample $u \leftarrow \mathbb{F}$, compute $v = y - u \cdot \mathsf{tag}$, and output $(u, v)$.

Namely, $\mathsf{Equivocate}(\mathsf{tag}, y)$ samples a random key $(u, v)$ conditioned on $h_{u,v}(\mathsf{tag}) = y$. Note that for any fixed $\mathsf{tag} \in \mathbb{F}$, we have that $h_{u,v}(\mathsf{tag})$ is uniform over $\mathbb{F}$ over the randomness of $(u, v)$. As a result, for all $\mathsf{tag} \in \mathbb{F}$, we have:

$$\Big(\mathsf{tag},\ (u,v) \leftarrow \mathbb{F}\times\mathbb{F},\ y = h_{u,v}(\mathsf{tag})\Big) \equiv \Big(\mathsf{tag},\ (u,v) \leftarrow \mathsf{Equivocate}(\mathsf{tag}, y),\ y \leftarrow \mathbb{F}\Big) \tag{1}$$

We now describe our first construction of a T-A**I**BO with lossiness $\ell = 1$. Let $t = t(\lambda)$ and $n = 3\lambda + t + 1$, $G : \{0,1\}^\lambda \to \{0,1\}^{3\lambda+t+1}$ be an injective PRG and $h$ be the pairwise independent hash function from above. We define the following algorithms:

– $\mathsf{InjectiveGen}(1^\lambda)$: Sample $(u, v) \leftarrow \mathbb{F} \times \mathbb{F}$ and set $\mathsf{fk} = (u, v)$.
– $\mathsf{LossyGen}(1^\lambda, \mathsf{tag}^*, x^*)$: Set $x_0^* = x^*$ and sample $x_1^* \leftarrow \{0,1\}^\lambda \setminus \{x_0^*\}$. Let $a = e \cdot (G(s_0^*) - G(s_1^*))^{-1}$. Compute $(u, v) \leftarrow \mathsf{Equivocate}(\mathsf{tag}^*, a)$ and output $\mathsf{fk} = (u, v)$.
– $F_{\mathsf{fk}}(\mathsf{tag}, s)$: Output $\mathsf{chop}(h_{u,v}(\mathsf{tag}) \cdot G(s))$.

**Claim 2.** *Suppose $G : \{0,1\}^\lambda \to \{0,1\}^{3\lambda+t+1}$ is an injective PRG. Then $(\mathsf{InjectiveGen}, \mathsf{LossyGen}, F)$ is a T-A**I**BO with input length $\lambda$, tag length $t$, output length $3\lambda+t$, and lossiness $\ell = 1$.*

*Proof.* We prove injectivity, 1-lossiness and indistinguishability.

Injectivity follows by the same argument as the TLF of Claim 1. In particular, an identical argument shows that, by injectivity of $G$, for any fixed $\mathsf{tag} \in \{0,1\}^t$, the probability that $F_{\mathsf{fk},\mathsf{tag}}$ is not injective is at most $\frac{2^{2\lambda+1}}{|\mathbb{F}|} = \frac{1}{2^{t+\lambda}}$. An union bound over the $2^t$ possible tags shows that the probability that $F_{\mathsf{fk},\mathsf{tag}}$ is not injective for some $\mathsf{tag}$ is at most $\frac{1}{2^\lambda}$.

The proof of 1-lossiness of Claim 1 directly extends here: $(\mathsf{fk}, F_k(x^*))$ can be generated as a randomized function of the unordered set $\{x_0^*, x_1^*\}$, where $x_0^*, x_1^* \leftarrow \{0,1\}^\lambda$ conditioned on $x_0^* \neq x_1^*$.

For indistinguishability, we first argue that for any $\mathsf{tag}^* \in \{0,1\}^t$ and any $x^* \in \{0,1\}^n$, the value $a$ sampled during $\mathsf{LossyGen}(1^\lambda, \mathsf{tag}^*, x^*)$ is computationally indistinguishable from uniformly random by PRG security of $G$, over the randomness of $x_1^*$. Then, indistinguishability follows by Eq. (1).

As for TLFs and T-A**L**BOs one can amplify lossiness by concatenating T-A**I**BOs evaluations on blocs of the input, which gives Theorem 2.

*Remark 1 (Generic Construction of T-A**I**BOs from TLFs).* In the above, we build a T-A**I**BO starting from the particular TLF from Section 4.1. We note that our transformation above can be made semi-generic, by assuming that the injective function keys $\mathsf{fk}$ generated by the $\mathsf{InjectiveGen}$ procedure of the base TLF are just uniformly random (as is the case in our construction). In that case, by mapping branches $\mathsf{tag}$ to function keys $\mathsf{fk}$ via a programmable pairwise independent hash function, we can generically get a T-A**I**BO from such a TLF, in the same way as above.

### 4.3 T-ALBOs

We now describe our construction of T-A**L**BOs. We prove the following theorems:

**Theorem 3 (T-ALBOs from Injective PRGs).** *Let $\ell = \ell(\lambda)$ and $t = t(\lambda)$ be any polynomials. Assuming the existence of injective PRGs, there exists a T-A**L**BO with input length $n = \ell\lambda$, tag length $t$, output length $m = \ell \cdot (3\lambda + t)$ and lossiness $\ell$.*

**Theorem 4 (Entropy-Preserving T-ALBOs from OWFs).** *Let $\ell = \ell(\lambda)$ and $t = t(\lambda)$ be any polynomials. Assuming the existence of one-way functions, there exists an entropy-preserving T-A**L**BO with input length $n = \ell\lambda$, tag length $t$, output length $m = \ell$ and lossiness $\ell$. In particular, such a T-A**L**BO is maximally entropy preserving.*

Again, we build on our construction of TLF from Section 4.1. We refer to our technical overview for a high level overview of the following construction.

*Construction.* Let $n = 3\lambda + t$. Let $\mathbb{F} = \mathbb{F}_{2^{3\lambda+t+1}}$. Let $G : \{0,1\}^n \to \{0,1\}^{2(n+1)}$ be a PRG. We will write $G(x) = (G^0(x), G^1(x))$. In particular, $G^0$ and $G^1$ are PRGs with input size $n$ and output size $n+1$; we will furthermore assume that $G^0$ and $G^1$ are injective. We define $(\mathsf{InjectiveGen}^0, \mathsf{LossyGen}^0, F^0)$ and $(\mathsf{InjectiveGen}^1, \mathsf{LossyGen}^1, F^1)$ as follows:

- $\mathsf{InjectiveGen}^b(1^\lambda)$: Sample $a \leftarrow \mathbb{F}$ and output $\mathsf{fk} = a$.
- $\mathsf{LossyGen}^b(1^\lambda, x^*, x_1^*)$: On input $x^*, x_1^* \in \{0,1\}^n$, set $x_0^* = x^*$. If $G^b(x_0^*) = G^b(x_1^*)$, output $\mathsf{fk} = 0$. Otherwise compute $a = e \cdot (G^b(x_0^*) - G^b(x_1^*))^{-1}$, and output $\mathsf{fk} = a$.
- $F^b_{\mathsf{fk}}(x) = \mathsf{chop}(a \cdot G^b(x)) \in \{0,1\}^n$.

Let $G' : \{0,1\}^\lambda \to \{0,1\}^n$ be a PRG. We build a T-A**L**BO $(\overline{\mathsf{InjectiveGen}}, \overline{\mathsf{LossyGen}}, \overline{F})$ as follows.

- $\overline{\mathsf{InjectiveGen}}(1^\lambda)$: For all $i \le t$ and $b \in \{0,1\}$, sample $\mathsf{fk}_i^b \leftarrow \mathsf{InjectiveGen}^b(1^\lambda)$, and output $\mathsf{fk} = \{\mathsf{fk}_i^b\}_{i\in[t],b\in\{0,1\}}$.
- $\overline{\mathsf{LossyGen}}(1^\lambda, \mathsf{tag}^*, x^*)$: Sample $x_1^* \leftarrow \{0,1\}^\lambda$, and set $x_0^{(0)} = G'(x^*)$, and $x_1^{(0)} = G'(x_1^*)$. For all $i \le t$, sample

$$\mathsf{fk}_i^{\mathsf{tag}_i^*} \leftarrow \mathsf{InjectiveGen}^{\mathsf{tag}_i^*}(1^\lambda).$$

  Set $x_0^{(i)} = F^{\mathsf{tag}_i^*}_{\mathsf{fk}_i^{\mathsf{tag}_i^*}} \circ \cdots \circ F^{\mathsf{tag}_i^*}_{\mathsf{fk}_1^{\mathsf{tag}_1^*}}(x_0) = F^{\mathsf{tag}_i^*}_{\mathsf{fk}_i^{\mathsf{tag}_i^*}}(x_0^{(i-1)})$ and $x_1^{(i)} = F^{\mathsf{tag}_i^*}_{\mathsf{fk}_i^{\mathsf{tag}_i^*}} \circ \cdots \circ F^{\mathsf{tag}_i^*}_{\mathsf{fk}_1^{\mathsf{tag}_1^*}}(x_1) = F^{\mathsf{tag}_i^*}_{\mathsf{fk}_i^{\mathsf{tag}_i^*}}(x_1^{(i-1)})$. Sample

$$\mathsf{fk}_i^{1-\mathsf{tag}_i^*} \leftarrow \mathsf{LossyGen}^{1-\mathsf{tag}_i^*}(1^\lambda, x_0^{(i)}, x_1^{(i)}),$$

  and output

$$\overline{\mathsf{fk}} = \{\mathsf{fk}_i^{(b)}\}_{i\in[t],b\in\{0,1\}}.$$

– $\overline{F}_{\mathsf{fk},\mathsf{tag}}(x)$: Output

$$y = F^{\mathsf{tag}_t}_{\mathsf{fk}_t^{(\mathsf{tag}_t)}} \circ \cdots \circ F^{\mathsf{tag}_1}_{\mathsf{fk}_1^{(\mathsf{tag}_1)}}(G'(x)).$$

**Claim 3.** *Suppose $G^0$, $G^1$ and $G'$ are injective PRGs, and $G = (G^0, G^1)$ is a PRG. Then $(\overline{\mathsf{InjectiveGen}}, \overline{\mathsf{LossyGen}}, \overline{F})$ is a T-A**L**BO with input length $\lambda$, tag length $t$, output length $3\lambda + t + 1$, and lossiness 1.*

*Proof.* We first show a useful property of $(\mathsf{InjectiveGen}^0, \mathsf{LossyGen}^0, F^0)$ and $(\mathsf{InjectiveGen}^1, \mathsf{LossyGen}^1, F^1)$.

– For all $x^* \in \{0,1\}^n$, we have the following computational indistinguishability:

$$(\mathsf{fk}^0_{inj}, F_{\mathsf{fk}^0}(x^*_1), x^*, \mathsf{fk}^1_{inj}) \overset{c}{\approx} (\mathsf{fk}^0_{inj}, u, x^*, \mathsf{fk}^1_{los}), \tag{2}$$

where $\mathsf{fk}^0_{inj} \leftarrow \mathsf{InjectiveGen}^0(1^\lambda)$, $\mathsf{fk}^1_{inj} \leftarrow \mathsf{InjectiveGen}^1(1^\lambda)$, $x^*_1 \leftarrow \{0,1\}^n$ and $\mathsf{fk}_{loss} \leftarrow \mathsf{LossyGen}^1(1^\lambda, \mathsf{tag}^*, x^*, x^*_1)$. Symmetrically, we have:

$$(\mathsf{fk}^1_{inj}, F_{\mathsf{fk}^1}(x^*_1), x^*, \mathsf{fk}^0_{inj}) \overset{c}{\approx} (\mathsf{fk}^1_{inj}, u, x^*, \mathsf{fk}^0_{los}), \tag{3}$$

where $\mathsf{fk}^1_{inj} \leftarrow \mathsf{InjectiveGen}^1(1^\lambda)$, $\mathsf{fk}^0_{inj} \leftarrow \mathsf{InjectiveGen}^0(1^\lambda)$, $x^*_1 \leftarrow \{0,1\}^n$ and $\mathsf{fk}_{loss} \leftarrow \mathsf{LossyGen}^0(1^\lambda, \mathsf{tag}^*, x^*, x^*_1)$.

These properties follow by PRG security of $G$, so that $F_{\mathsf{fk}^b_{inj}}(x^*_1)$ is computationally indistinguishable from uniformly random over the randomness of $G^b(x^*)$, while indistinguishability of $\mathsf{fk}^{1-b}_{inj}$ and $\mathsf{fk}^{1-b}_{los}$ follows over the (independent) randomness of $G^{1-b}(x^*)$.

We now prove that the construction above is a T-A**L**BO.

*Injectivity.* Fix $x_0 \neq x_1 \in \{0,1\}^\lambda$. By injectivity of $G'$, we have $G'(x_0) \neq G'(x_1)$. Let $i \in [t]$ and $b \in \{0,1\}$, and let $x^{(i)}_0 \neq x^{(i)}_1 \in \{0,1\}^n$. By injectivity of $G^b$, the probability over $\mathsf{fk}^b_i \leftarrow \mathsf{InjectiveGen}^b(1^\lambda)$ that $F^b_{\mathsf{fk}_i^{(b)}}(x^{(i)}_0) = F^b_{\mathsf{fk}_i^{(b)}}(x^{(i)}_1)$ is $2/|\mathbb{F}|$ (which correspond to either $a = e \cdot (G^b(x^{(i)}_0) - G^b(x^{(i)}_1)^{-1}$ or $a = 0$). In particular, for any fixed $\mathsf{tag} \in \{0,1\}^t$, we have, by taking an union bound over $i \in [t]$, that the probability over $\mathsf{fk} \leftarrow \mathsf{InjectiveGen}(1^\lambda)$ that $F_{\mathsf{fk},\mathsf{tag}}(G'(x_0)) = F_{\mathsf{fk},\mathsf{tag}}(G'(x_1))$ is at most $2t/|\mathbb{F}|$. Then, by union bound over all $\mathsf{tag} \in \{0,1\}^t$ and pairs of input $x_0 \neq x_1 \in \{0,1\}^\lambda$, the probability that there exists a tag $\mathsf{tag}$ and two inputs $x_0 \neq x_1$ such that $F_{\mathsf{fk},\mathsf{tag}}(G'(x_0)) = F_{\mathsf{fk},\mathsf{tag}}(G'(x_1))$ is at most $\frac{2t \cdot 2^t \cdot 2^{2\lambda}}{|\mathbb{F}|} = \frac{t}{2^\lambda}$, which is negligible.

An almost identical argument (without the union bound over all tags) shows that the branch $\mathsf{tag}^*$ is injective in lossy mode.

*1-Lossiness.* If $\mathsf{LossyGen}$ samples $x^*_1 = x^*$, then $F^{1-\mathsf{tag}^*_i}_{\mathsf{fk}_i^{1-\mathsf{tag}^*_i}}(\cdot)$ is the zero function for all $i \in [t]$, so that $(\mathsf{fk}, \{F_{\mathsf{fk},\mathsf{tag}}(x^*)\}_{\mathsf{tag} \neq \mathsf{tag}^*})$ is independent of $x^*$. In particular $H_\infty(x^* \mid \mathsf{fk}, \{F_{\mathsf{fk},\mathsf{tag}}(x^*)\}_{\mathsf{tag} \neq \mathsf{tag}^*}, x^*_1 = x^*) = n$.

Otherwise, let $\mathsf{tag} \neq \mathsf{tag}^*$. Let $i$ be the smallest index such that $\mathsf{tag}_i \neq \mathsf{tag}^*_i$. By construction, we have

$$F^{\mathsf{tag}_i}_{\mathsf{fk}_i^{(\mathsf{tag}_i)}} \circ \cdots \circ F^{\mathsf{tag}_1}_{\mathsf{fk}_1^{(\mathsf{tag}_1)}}(G'(x^*)) = F^{\mathsf{tag}_i}_{\mathsf{fk}_i^{(\mathsf{tag}_i)}} \circ \cdots \circ F^{\mathsf{tag}_1}_{\mathsf{fk}_1^{(\mathsf{tag}_1)}}(G'(x_1^*)),$$

and therefore $F_{\mathsf{fk},\mathsf{tag}}(x^*) = F_{\mathsf{fk},\mathsf{tag}}(x_1^*)$. In particular, the views $(x^*, \mathsf{fk}, \{F_{\mathsf{fk},\mathsf{tag}}(x^*)\}_{\mathsf{tag}\neq\mathsf{tag}^*})$ and $(x_1^*, \mathsf{fk}, \{F_{\mathsf{fk},\mathsf{tag}}(x_1^*)\}_{\mathsf{tag}\neq\mathsf{tag}^*})$ are identical and therefore $H_\infty(x^* \mid \mathsf{fk}, \{F_{\mathsf{fk},\mathsf{tag}}(x^*)\}_{\mathsf{tag}\neq\mathsf{tag}^*}, x_1^* \neq x^*) \geq 1$.

*Indistinguishability.* First, $x_0^{(0)}$ and $x_1^{(0)}$ look pseudorandom by security of $G^0$. Then, we switch injective keys to lossy keys, one by one, using our special TLF property.

*Hybrid $H_j$* : We switch how we generate $\mathsf{fk}$. We sample $x_1^* \in \{0,1\}^n$, and set $x_0^{(0)} = G'(x^*)$, and $x_1^{(0)} \leftarrow \{0,1\}^n$. For all $i \leq t$, sample

$$\mathsf{fk}_i^{\mathsf{tag}_i^*} \leftarrow \mathsf{InjectiveGen}^{\mathsf{tag}_i^*}(1^\lambda).$$

For $i > j$, sample $\mathsf{fk}_i^{1-\mathsf{tag}_i^*} \leftarrow \mathsf{InjectiveGen}^{1-\mathsf{tag}_j^*}(1^\lambda)$.

Set $x_0^{(i)} = F_{\mathsf{fk}_i^{1-\mathsf{tag}_i^*}} \circ \cdots \circ F_{\mathsf{fk}_1^{1-\mathsf{tag}_1^*}}(x_0)$ and set for $i \leq j$, $x_1^{(i)} \leftarrow \{0,1\}^n$. For $i \leq j$, sample

$$\mathsf{fk}_i^{\mathsf{tag}_i^*} \leftarrow \mathsf{LossyGen}^{\mathsf{tag}_i^*}(1^\lambda, \mathsf{tag}^*, x_0^{(1^\lambda, \mathsf{tag}^*,)}, x_1^{(i)}),$$

and output

$$\overline{\mathsf{fk}} = \{\mathsf{fk}_i^{(b)}\}_{i\leq t, b\in\{0,1\}}.$$

The distribution of $(\mathsf{tag}^*, x^*, \mathsf{fk}, G'(x_1^*))$ is indistinguishable from $(\mathsf{tag}^*, x^*, \mathsf{fk}, u)$ where $u \leftarrow \{0,1\}^n$ by security of $G'$, regardless of how $\mathsf{fk}$ is generated. In particular, the distribution $(\mathsf{tag}^*, x^*, \mathsf{fk})$ generated $H_0$ is computationally indistinguishable from when $\mathsf{fk}$ is generated using $\mathsf{InjectiveGen}$, and $(\mathsf{tag}^*, x^*, \mathsf{fk})$ in $H_t$ is indistinguishable from when $\mathsf{fk}$ is generated using $\mathsf{LossyGen}(x^*)$.

It remains to argue that $H_j$ and $H_{j+1}$ are indistinguishable for all $j \leq t$. But this follows from our special joint indistinguishability property of $(\mathsf{InjectiveGen}^0, \mathsf{LossyGen}^0, F^0)$ and $(\mathsf{InjectiveGen}^1, \mathsf{LossyGen}^1, F^1)$ (Eq. (2), Eq. (3)), as long as $x_0^{(k)} \neq x_1^{(k)}$ for all $k \leq j$, which happens with probability at least $1 - t/2^m$.

*Amplifying Lossiness.* As in our construction of TLF, the construction above of T-A**L**BO only has lossiness 1. We note that we can also amplify lossiness for T-A**L**BOs, which gives Theorem 3.

We now move on to our construction of (maximally) entropy-preserving T-A**L**BO.

*Entropy-Preserving T-A**L**BOs.* Next, we describe how to obtain a (maximally) *entropy-preserving* T-A**L**BOs with output size $m$. We first start by giving a construction of an entropy-preserving T-A**L**BOs with output size 1 and lossiness 1.

Our starting point is our previous construction of a T-A**L**BO. Let $\overline{\mathsf{chop}} : \{0,1\}^n \to \{0,1\}$ be the function that outputs the *last* bit of its input. In particular $\mathsf{chop}(x+e) \neq \mathsf{chop}(x)$ for all $x \in \{0,1\}^n$. We'll slightly modify our previous construction as follows. First, we will further chop the final output of the previous construction to match the output size $m = 1$. Second, we will slightly modify the $\mathsf{LossyGen}$ algorithm to resample a fresh key if $x^*$ and $x_1^*$ collide. The reason is that entropy-preserving would otherwise only hold with overwhelming probability in lossy mode (over the choice of the function key).

- $\widetilde{\mathsf{InjectiveGen}}(1^\lambda)$: Sample $\mathsf{fk} \leftarrow \overline{\mathsf{InjectiveGen}}(1^\lambda)$. Sample $a \leftarrow \mathbb{F}$ and output $(\mathsf{fk}, a)$.
- $\widetilde{\mathsf{LossyGen}}(1^\lambda, \mathsf{tag}^*, x^*)$: Sample $\mathsf{fk} \leftarrow \overline{\mathsf{LossyGen}}(1^\lambda, \mathsf{tag}^*, x^*)$, which also internally samples $x_1^* \leftarrow \{0,1\}^\lambda$. If $\overline{F}_{\mathsf{fk},\mathsf{tag}^*}(x^*) = \overline{F}_{\mathsf{fk},\mathsf{tag}^*}(x_1^*)$, repeat the above. Otherwise, compute $y^* = \overline{F}_{\mathsf{fk},\mathsf{tag}^*}(x^*)$ and $y_1 = \overline{F}_{\mathsf{fk},\mathsf{tag}^*}(x_1^*)$. Set $a = e \cdot (y^* - y_1)^{-1}$. Output $(\mathsf{fk}, a)$.
- $\widetilde{F}_{\mathsf{fk},\mathsf{tag}}(x)$: Output
$$y = \overline{\mathsf{chop}}(a \cdot \tilde{F}_{\mathsf{fk},\mathsf{tag}}(x)),$$
where $a \leftarrow \mathbb{F}$ is sampled during key generation.

**Claim 4.** *Let* $(\overline{\mathsf{InjectiveGen}}, \overline{\mathsf{LossyGen}}, \overline{F})$ *be the T-A**L**BO of Claim 3. Then* $(\widetilde{\mathsf{InjectiveGen}}, \widetilde{\mathsf{LossyGen}}, \widetilde{F})$ *is an entropy-preserving T-A**I**BO with input length* $\lambda$*, tag length* $t$*, output length* $1$*, and lossiness* $\ell = 1$*.*

For indistinguishability, we first show that the resampling in $\widetilde{\mathsf{LossyGen}}$ only happens with negligible probability. Indeed, such an event occurs only if $x^* = x_1^*$, or if there exists some $i \in [t]$ such that $F^{\mathsf{tag}_i^*}(x_0^{(i-1)}) = F^{\mathsf{tag}_i^*}(x_1^{(i-1)})$ (using the notation of Claim 3).

If $i^*$ is the smallest such $i$, then $x_1^{(i^*-1)}$ is computationally indistinguishable from independently uniform in $\{0,1\}^n$ even given $x^*$ and $\mathsf{fk}$ (which we showed in the proof of 1-lossiness of the T-A**L**BO). Therefore $F^{\mathsf{tag}_i^*}(x_1^{(i-1)})$ is distributed (computationally close to) uniformly random in $\{0,1\}^n$ even given $x^*$ and $\mathsf{fk}$ (this follows by our special properties of $F^0$ and $F^1$), which implies that $F^{\mathsf{tag}_i^*}(x_0^{(i-1)}) = F^{\mathsf{tag}_i^*}(x_1^{(i-1)})$ with at most negligible probability. Our claim follows using an union bound over $i \in [t]$, and indistinguishability then follows by indistinguishability from Claim 3.

For lossiness, we have, as previously, that the views $(x^*, \mathsf{fk}, \{F_{\mathsf{fk},\mathsf{tag}}(x^*)\}_{\mathsf{tag} \neq \mathsf{tag}^*})$ and $(x_1^*, \mathsf{fk}, \{F_{\mathsf{fk},\mathsf{tag}}(x_1^*)\}_{\mathsf{tag} \neq \mathsf{tag}^*})$ are identical for any fixed $\mathsf{tag}^* \in \{0,1\}^t$, where $x^* \leftarrow \{0,1\}^n$, $\mathsf{fk} \leftarrow \mathsf{LossyGen}(1^\lambda, \mathsf{tag}^*, x^*)$ and $x_1^*$ is internally sampled by $\mathsf{LossyGen}$. Now by construction, we have $F_{\mathsf{fk},\mathsf{tag}^*}(x^*) \neq F_{\mathsf{fk},\mathsf{tag}^*}(x_1^*)$ and therefore
$$H_\infty(F_{\mathsf{fk}}(\mathsf{tag}^*, x^*) \mid \mathsf{fk}, \{F_{\mathsf{fk},\mathsf{tag}}(x^*) \ : \ \mathsf{tag} \neq \mathsf{tag}^*\}) \geq 1.$$

Note that the only property that used the injectivity of the PRGs in 3 is the injectivity of the T-A**L**BO. As we do not require injectivity for entropy-preserving T-A**L**BOs (the output size being shorter than the input size anyway), we can instantiate the PRGs $G$ and $G'$ with standard (non-necessarily injective) PRGs, and thus we obtain an entropy-preserving T-A**L**BO from any one-way function.

*Amplifying Lossiness.* Again, we can amplify the lossiness $\ell$ by concatenating images of the previous entropy-preserving T-A**L**BO on blocs of the input, which gives Theorem 4.

## 5 Applications of T-ALBOs

We first recall in Section 5.1 the definition of pseudo-entropy functions (PEF) introduced by [BHK11]. We note that any entropy-preserving T-A**L**BO directly gives such a PEF, thus giving a construction from one-way functions (Theorem 5). Then, we describe applications of PEF, by constructing (1) extractor-dependent extractors (Theorem 6), (2) leakage-resilient, deterministic MACs (Theorem 15), and (3) leakage-resilient, public-coin symmetric encryption schemes (Theorem 16).

### 5.1 Pseudo-Entropy Functions.

**Definition 5 (Pseudo-Entropy Functions.).** *A pseudo-entropy function family with input length $n = n(\lambda)$, output length $m = m(\lambda)$, and lossiness parameter $\ell = \ell(\lambda)$ consists of the following PPT algorithms* $(\mathsf{Gen}, \mathsf{LossyGen}, f)$*:*

- $k \leftarrow \mathsf{Gen}(1^\lambda)$*: generates a key $k$.*
- $k \leftarrow \mathsf{LossyGen}(1^\lambda, x^*)$*: on input $x^* \in \{0,1\}^n$, outputs a key $k$.*
- $y = f_k(x)$*: on input $x \in \{0,1\}^n$, deterministically outputs $y \in \{0,1\}^m$.*

*We require the following properties:*

*$\ell$-Lossiness: For all $x^* \in \{0,1\}^n$:*

$$H_\infty\left(f_k(x^*)\right) \,|\, \{f_k(x)\}_{x \neq x'}) \geq \ell$$

*over the randomness of $k \leftarrow \mathsf{LossyGen}(1^\lambda, x^*)$.*

*Indistinguishability: For all $x^* \in \{0,1\}^n$:*

$$\mathsf{Gen}(1^\lambda) \overset{c}{\approx} \mathsf{LossyGen}(1^\lambda, x^*).$$

Next, we show the following:

**Theorem 5 (PEFs from OWFs).** *Let $\ell = \ell(\lambda)$ and $t = t(\lambda)$ be polynomials. Assuming the existence of one-way functions, there exists a PEF with input length $t$, output length $\ell$ and lossiness $\ell$.*

We refer to Appendix A.1 for construction and proof of Theorem 5.

### 5.2 Extractor-Dependent Extractors

We show how to build ED-extractors with auxiliary information from one-way functions:

**Theorem 6 (ED-Extractors from OWFs).** *Assuming the existence of one-way functions there exists an ED-extractor for $\alpha$-entropy sources with auxiliary information, where $\alpha = \lambda^{\Omega(1)}$.*

We refer to Appendix A.2 for the definition of ED-extractors, our construction and the proof of Theorem 6.

### 5.3 Leakage-Resilient Symmetric-Key MACs

We note here, as already observed by [BHK11], that PEFs with super-logarithmic lossiness $\ell$ directly give deterministic, leakage-resilient MACs with selective unforgeability security. Leakage resilience here denotes the fact that the adversary can learn any efficiently computable function of the secret key, as long as the output size of the function is bounded by some leakage-size parameter $L$. Selective unforgeability security means that the adversary chooses the message $m^*$ on which it will produce a forgery ahead of time, before seeing the leakage or seeing any authentication tags for chosen messages. We refer the reader to [HLWW13] for a formal definition of leakage-resilient MACs.

**Theorem 7 (Deterministic Leakage-Resilient MACs from OWFs).** *Let $m = \text{poly}(\lambda)$ be a message length and $t \geq \omega(\log \lambda)$ be a tag length. Assuming one-way functions, there exists, for all $L$ such that $t - L = \omega(\log \lambda)$, a deterministic MAC with message length $m$ and tag length $t$ that satisfies selective unforgeability even given leakage of size $L$.*

We refer to Appendix A.3 for the construction and proof of Theorem 7.

### 5.4 Leakage-Resilient Symmetric Encryption

Next, we remark that any PEF also yields a leakage-resilient symmetric encryption scheme. In a leakage-resilient scheme, the adversary can get some leakage of the secret key, potentially after making some CPA-encryption queries but before seeing the challenge ciphertext. As in the case of MACs, we assume the leakage consists of an arbitrary efficiently computable function applied to the secret key, as long as the outut size of the function is at most $L$ bits for some leakage parameter $L$. Our scheme is public-coin, meaning that all the internal randomness of the encryption procedure is explicitly contained in the ciphertext. We refer the reader to [HLWW13] for a formal definition of leakage-resilient symmetric-key encryption.

**Theorem 8 (Leakage-Resilient Symmetric Encryption from OWFs).** *For any polynomial leakage amount $L = L(\lambda)$ and message length $m = m(\lambda)$, assuming one-way functions exists, there exists a public-coin, symmetric encryption scheme with message length $m$, ciphertext size $m + O(\lambda)$, and key size $O((L + \lambda)\lambda)$.*

We refer to Appendix A.4 for the construction and proof of Theorem 8.

### 5.5 Symmetric-Key Encryption Secure against Selective Opening Attacks

Finally, we show that PEFs have applications to security against selective opening attacks.

**Theorem 9 (Selective Opening Security from OWFs).** *Assuming one-way functions exist, there exists a symmetric-key encryption scheme that achieves simulation-security against selective opening of keys and randomness (Definition 10).*

We refer to Appendix A.5 for a definition of selective opening security, and the construction and proof of Theorem 9.

## 6 Application of T-AIBOs to CCA Security

We prove the following theorem:

**Theorem 10 (CCA Encryption from Strong Trapdoor Functions).** *Let* $d = d(\lambda)$, $n = n(\lambda) = \omega(\log \lambda)$, $\rho = d \cdot n$, *and* $m = \max(n+1, \lambda)$. *Let* $\mathsf{TDF}$ *be a trapdoor function with input length* $\rho$. *Suppose that no time* $T = 2^n \cdot \mathrm{poly}(\lambda)$ *adversary can invert* $\mathsf{TDF}$ *with probability* $\frac{2^d}{2^\rho} \cdot \epsilon$ *for any non-negligible* $\epsilon$. *Assume furthermore the existence of an injective PRG* $G : \{0,1\}^n \to \{0,1\}^m$.

*Then there exists a CCA-secure (public-key) encryption scheme.*

We refer to Appendix B for the construction and proof of Theorem 10.

## References


ADW09. Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 36–54. Springer, Heidelberg, August 2009.

AGV09. Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In Omer Reingold, editor, *TCC 2009: 6th Theory of Cryptography Conference*, volume 5444 of *Lecture Notes in Computer Science*, pages 474–495. Springer, Heidelberg, March 2009.

BBN+09. Mihir Bellare, Zvika Brakerski, Moni Naor, Thomas Ristenpart, Gil Segev, Hovav Shacham, and Scott Yilek. Hedged public-key encryption: How to protect against bad randomness. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 232–249. Springer, Heidelberg, December 2009.

BDWY12. Mihir Bellare, Rafael Dowsley, Brent Waters, and Scott Yilek. Standard security does not imply security against selective-opening. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 645–662. Springer, Heidelberg, April 2012.

BFO08. Alexandra Boldyreva, Serge Fehr, and Adam O'Neill. On notions of security for deterministic encryption, and efficient constructions without random oracles. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 335–359. Springer, Heidelberg, August 2008.

BGI15. Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 337–367. Springer, Heidelberg, April 2015.

BHK11. Mark Braverman, Avinatan Hassidim, and Yael Tauman Kalai. Leaky pseudo-entropy functions. In Bernard Chazelle, editor, *ICS 2011: 2nd Innovations in Computer Science*, pages 353–366. Tsinghua University Press, January 2011.

BHY09. Mihir Bellare, Dennis Hofheinz, and Scott Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In Antoine Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 1–35. Springer, Heidelberg, April 2009.

CG88. Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM J. Comput.*, 17(2):230–261, 1988.

CPW20. Suvradip Chakraborty, Manoj Prabhakaran, and Daniel Wichs. Witness maps and applications. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020: 23rd International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 12110 of *Lecture Notes in Computer Science*, pages 220–246. Springer, Heidelberg, May 2020.

DGK17. Yevgeniy Dodis, Siyao Guo, and Jonathan Katz. Fixing cracks in the concrete: Random oracles with auxiliary input, revisited. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part II*, volume 10211 of *Lecture Notes in Computer Science*, pages 473–495. Springer, Heidelberg, April / May 2017.

DHLW10. Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Cryptography against continuous memory attacks. In *51st Annual Symposium on Foundations of Computer Science*, pages 511–520. IEEE Computer Society Press, October 2010.

DNRS99. Cynthia Dwork, Moni Naor, Omer Reingold, and Larry J. Stockmeyer. Magic functions. In *40th Annual Symposium on Foundations of Computer Science*, pages 523–534. IEEE Computer Society Press, October 1999.

DORS08. Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam D. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.

DVW20. Yevgeniy Dodis, Vinod Vaikuntanathan, and Daniel Wichs. Extracting randomness from extractor-dependent sources. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part I*, vol-

ume 12105 of *Lecture Notes in Computer Science*, pages 313–342. Springer, Heidelberg, May 2020.

FGK+13. David Mandell Freeman, Oded Goldreich, Eike Kiltz, Alon Rosen, and Gil Segev. More constructions of lossy and correlation-secure trapdoor functions. *Journal of Cryptology*, 26(1):39–74, January 2013.

FHKW10. Serge Fehr, Dennis Hofheinz, Eike Kiltz, and Hoeteck Wee. Encryption schemes secure against chosen-ciphertext selective opening attacks. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 381–402. Springer, Heidelberg, May / June 2010.

GGH19. Sanjam Garg, Romain Gay, and Mohammad Hajiabadi. New techniques for efficient trapdoor functions and applications. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part III*, volume 11478 of *Lecture Notes in Computer Science*, pages 33–63. Springer, Heidelberg, May 2019.

GGM86. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, October 1986.

GI14. Niv Gilboa and Yuval Ishai. Distributed point functions and their applications. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 640–658. Springer, Heidelberg, May 2014.

GKK20. Ankit Garg, Yael Tauman Kalai, and Dakshita Khurana. Low error efficient computational extractors in the CRS model. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 373–402. Springer, Heidelberg, May 2020.

GL89. Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *21st Annual ACM Symposium on Theory of Computing*, pages 25–32. ACM Press, May 1989.

GUV07. V. Guruswami, C. Umans, and S. Vadhan. Unbalanced expanders and randomness extractors from parvaresh-vardy codes. In *Twenty-Second Annual IEEE Conference on Computational Complexity (CCC'07)*, pages 96–108, 2007.

HJO+16. Brett Hemenway, Zahra Jafargholi, Rafail Ostrovsky, Alessandra Scafuro, and Daniel Wichs. Adaptively secure garbled circuits from one-way functions. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part III*, volume 9816 of *Lecture Notes in Computer Science*, pages 149–178. Springer, Heidelberg, August 2016.

HKW20. Susan Hohenberger, Venkata Koppula, and Brent Waters. Chosen ciphertext security from injective trapdoor functions. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 836–866. Springer, Heidelberg, August 2020.

HLOV11. Brett Hemenway, Benoît Libert, Rafail Ostrovsky, and Damien Vergnaud. Lossy encryption: Constructions from general assumptions and efficient selective opening chosen ciphertext security. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 70–88. Springer, Heidelberg, December 2011.

HLWW13. Carmit Hazay, Adriana López-Alt, Hoeteck Wee, and Daniel Wichs. Leakage-resilient cryptography from minimal assumptions. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 160–176. Springer, Heidelberg, May 2013.

Hof12. Dennis Hofheinz. All-but-many lossy trapdoor functions. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 209–227. Springer, Heidelberg, April 2012.

HPW15. Carmit Hazay, Arpita Patra, and Bogdan Warinschi. Selective opening security for receivers. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology – ASIACRYPT 2015, Part I*, volume 9452 of *Lecture Notes in Computer Science*, pages 443–469. Springer, Heidelberg, November / December 2015.

HR14. Dennis Hofheinz and Andy Rupp. Standard versus selective opening security: Separation and equivalence results. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 591–615. Springer, Heidelberg, February 2014.

HRW16. Dennis Hofheinz, Vanishree Rao, and Daniel Wichs. Standard security does not imply indistinguishability under selective opening. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B: 14th Theory of Cryptography Conference, Part II*, volume 9986 of *Lecture Notes in Computer Science*, pages 121–145. Springer, Heidelberg, October / November 2016.

KOS10. Eike Kiltz, Adam O'Neill, and Adam Smith. Instantiability of RSA-OAEP under chosen-plaintext attack. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 295–313. Springer, Heidelberg, August 2010.

MW20. Tal Moran and Daniel Wichs. Incompressible encodings. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 494–523. Springer, Heidelberg, August 2020.

NS09. Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 18–35. Springer, Heidelberg, August 2009.

PW08. Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 187–196. ACM Press, May 2008.

Raz05. Ran Raz. Extractors with weak random seeds. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 11–20. ACM Press, May 2005.

Zha16. Mark Zhandry. The magic of ELFs. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 479–508. Springer, Heidelberg, August 2016.

# SUPPLEMENTARY MATERIAL

## A Applications of T-ALBOs

We first recall in Appendix A.1 the definition of pseudo-entropy functions (PEF) introduced by [BHK11]. We note that any entropy-preserving T-A**L**BO directly gives such a PEF, thus giving a construction from one-way functions (Theorem 11). Then, we describe applications of PEF, by constructing (1) extractor-dependent extractors (Theorem 12), (2) leakage-resilient, deterministic MACs (Theorem 15), and (3) leakage-resilient, public-coin symmetric encryption schemes (Theorem 16).

### A.1 Pseudo-Entropy Functions.

**Definition 6 (Pseudo-Entropy Functions.).** *A pseudo-entropy function family with input length $n = n(\lambda)$, output length $m = m(\lambda)$, and lossiness parameter $\ell = \ell(\lambda)$ consists of the following PPT algorithms* $(\mathsf{Gen}, \mathsf{LossyGen}, f)$*:*

- $k \leftarrow \mathsf{Gen}(1^\lambda)$*: generates a key $k$.*
- $k \leftarrow \mathsf{LossyGen}(1^\lambda, x^*)$*: on input $x^* \in \{0,1\}^n$, outputs a key $k$.*
- $y = f_k(x)$*: on input $x \in \{0,1\}^n$, deterministically outputs $y \in \{0,1\}^m$.*

*We require the following properties:*

*$\ell$-Lossiness: For all $x^* \in \{0,1\}^n$:*

$$H_\infty\left(f_k(x^*)\right) \mid \{f_k(x)\}_{x \neq x'}) \geq \ell$$

*over the randomness of $k \leftarrow \mathsf{LossyGen}(1^\lambda, x^*)$.*

*Indistinguishability: For all $x^* \in \{0,1\}^n$:*

$$\mathsf{Gen}(1^\lambda) \overset{c}{\approx} \mathsf{LossyGen}(1^\lambda, x^*).$$

Next, we show the following:

**Theorem 11 (PEFs from OWFs).** *Let $\ell = \ell(\lambda)$ and $t = t(\lambda)$ be polynomials. Assuming the existence of one-way functions, there exists a PEF with input length $t$, output length $\ell$, lossiness $\ell$ and key length $\ell\lambda$.*

Next, we note that PEFs are directly implied by any T-A**L**BO.

**Claim 5.** *Suppose* $(\mathsf{InjectiveGen}, \mathsf{LossyGen}, F)$ *is an entropy-preserving T-A**L**BO with input length $n$, output length $m$, tag length $t$ and lossiness parameter $\ell$. Then there exists a PEF with input length $t$, output length $m$, key size $n$ and lossiness parameter $\ell$.*

*Proof.* Let $(\mathsf{InjectiveGen}, \mathsf{LossyGen}, F)$ be such a T-A**L**BO. We define a PEF $(\mathsf{Gen}, \mathsf{LossyGen}, f)$ as follows:

- $\mathsf{Gen}(1^\lambda)$: Sample a uniformly random $s \in \{0,1\}^n$, and computes $\mathsf{fk} \leftarrow \mathsf{InjectiveGen}$. Output $k = (\mathsf{fk}, s)$.
- $\mathsf{LossyGen}(1^\lambda, x^*)$: Sample $s^* \leftarrow \{0,1\}^n$, and compute $\mathsf{fk} \leftarrow \mathsf{LossyGen}(x^*, s^*)$, interpreting $x^*$ as a tag for the T-A**L**BO and $s^*$ in its input space. Output $k = (\mathsf{fk}, s^*)$.
- $f_k(x)$: On input $x \in \{0,1\}^t$ and a key $k = (\mathsf{fk}, s)$, output $F_{\mathsf{fk},x}(s)$ (again interpreting $x$ as a tag for the T-A**L**BO, and $s$ as an input to the T-A**L**BO).

Indistinguishability follows immediately by indistinguishability of the T-A**L**BO. For lossiness, if $k^* \leftarrow \mathsf{LossyGen}(x^*)$, then

$$\begin{aligned} H_\infty\left(f_k(x^*)\right) \mid \{f_k(x)\}_{x \neq x'}\right) &= H_\infty\left(F_{\mathsf{fk},x^*}(s^*)\right) \mid \{F_{\mathsf{fk},x^*}(s^*)\}_{x \neq x^*}\right) \\ &\geq H_\infty\left(F_{\mathsf{fk},x^*}(s^*)\right) \mid \mathsf{fk}\,,\, \{F_{\mathsf{fk},x^*}(s^*)\}_{x \neq x^*}\right) \\ &\geq \ell, \end{aligned}$$

where the last equality is due to the entropy-preserving property of the T-A**L**BO.

Theorem 11 follows by combining the above with Theorem 4.

### A.2 Extractors for Extractor-Dependent Sources

We first recall the definition of extractor-dependent source extractors (ED-extractors), defined in [DVW20]. The following is taken verbatim from [DVW20].

**Definition 7 (Extractor-Dependent Source Extraction).** *An extractor for $\alpha$-entropy extractor-dependent sources ($\alpha$-ED-Extractor) consists of two polynomial-time algorithms* $(\mathsf{SeedGen}, \mathsf{EDExt})$ *with the following syntax:*

- $\mathsf{seed} \leftarrow \mathsf{SeedGen}(1^\lambda)$ *is a randomized algorithm that generates* $\mathsf{seed}$.
- $\mathsf{EDExt}(s, \mathsf{seed})$ *is a deterministic algorithm that takes a sample $s \in \{0,1\}^n$, together with* $\mathsf{seed}$ *and outputs a value $y \in \{0,1\}^m$ for some polynomial length parameters $n = n(\lambda), m = m(\lambda)$.*

*Consider an adversarial source/distinguisher pair $(\mathcal{D}, \mathcal{S})$ and define the following extraction experiment* $\mathsf{EDGame}^{\mathcal{D},\mathcal{S}}(1^\lambda)$*:*

- *Sample a random bit $b \leftarrow \{0,1\}$ and a random* $\mathsf{seed} \leftarrow \mathsf{SeedGen}(1^\lambda)$.
- *Run* $(s, \mathsf{aux}) \leftarrow \mathcal{D}^{\mathsf{EDExt}(\cdot,\mathsf{seed})}(1^\lambda)$.
- *If $b = 0$ set $r = \mathsf{EDExt}(x, \mathsf{seed})$ else if $b = 1$ set $r \leftarrow \{0,1\}^m$.*
- *Let* $b' = \mathcal{S}(1^\lambda, \mathsf{seed}, \mathsf{aux}, r)$.

*We say that $\mathcal{D}$ is an $\alpha$-legal extractor-dependent source if the following conditions hold:*

1. *The probability that $\mathcal{D}$ queries its oracle on the value $x$ that it outputs is negligible.*

2. $H_\infty(X|\mathsf{AUX}, \mathsf{SEED}) \geq \alpha(\lambda)$, *where* $X, \mathsf{SEED}, \mathsf{AUX}$ *denotes the joint distribution of the values* $x, \mathsf{seed}, \mathsf{aux}$ *in the above experiment.*

*An* $\alpha$*-ED-Extractor is secure if for all* $\alpha$*-legal polynomial-time sources* $\mathcal{D}$ *and all polynomial-time distinguishers* $\mathcal{S}$*, the above experiment satisfies*

$$\left|\Pr[b = b'] - \frac{1}{2}\right| = \mathrm{negl}(\lambda).$$

The rest of the section is dedicated to prove the following theorem:

**Theorem 12 (ED-Extractors from OWFs).** *Assuming the existence of one-way functions there exists an ED-extractor for* $\alpha$*-entropy sources with auxiliary information, where* $\alpha = \lambda^{\Omega(1)}$.

We recall the definition of a 2-source extractor, and a construction due to Raz [Raz05].

**Definition 8 ((Strong, Average-Case) Two-Source Extractor [CG88]).** *We say that an efficient function* $2\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$ *is an* $(e_1, e_2, \delta)$-strong 2-source extractor *if for all random variables* $(X_1, X_2, Z)$ *such that* $X_1, X_2$ *are independent conditioned on* $Z$ *and* $H_\infty(X_1|Z) \geq e_1, H_\infty(X_2|Z) \geq e_2$ *we have* $\mathsf{SD}(\ (Z, X_2, 2\mathsf{Ext}(X_1; X_2))\ ,\ (Z, X_2, U_m)) \leq \delta$ *where* $U_m$ *is a uniformly string of length* $m$.

**Theorem 13 ( [Raz05]).** *For any polynomial input length* $n = \mathrm{poly}(\lambda)$*, any* $e_1 = \lambda^{\Omega(1)}$ *and any* $e_2 = (1/2 + \Omega(1))n$*, there exist* $(e_1, e_2, \delta)$*-extractor with input length* $n$*, output length* $m = \lambda^{\Omega(1)}$ *and error* $\delta = 2^{-\lambda^{\Omega(1)}}$.

Next, we present a construction of an ED-extractor, starting from PEFs. Our construction essentially just abstracts out the construction of [DVW20], which relied on (non-targeted) all-lossy-but-one functions, in terms of PEFs. Doing so essentially shows that the construction [DVW20] only needs a *targeted* form of all-lossy-but-one functions.

*Construction Outline.* The basic idea is to set the seed $\mathsf{seed} = k$ to consists of a PEF key $k$ for a PEF $f_k$. The extractor on input $x$ "hashes" $x$ to some much smaller value $z$ and computes $y = f_k(z)$ to to be the PEF output. We then think of $y$ as a seed to a standard seeded randomness extractor and output the extracted randomness $r = \mathsf{Ext}(x; y)$. The idea behind the proof of security is to "guess" the value $z$ that the sample $x$ chosen by the source will hash to and select the PEF key $K$ to preserve entropy on $z$. This is indistinguishable even if the adversary sees the seed later. The above change ensures that $y = f_k(z)$ is uniformly random and independent of all the values $y_i = f_k(z_i)$ that were used to compute prior extractor outputs. Therefore, this essentially corresponds to using a completely fresh and independent seed $y$ when extracting from the challenge sample $x$, and guarantees that the extracted output looks random.

The above relies on hashing, but we can replace the hash by a standard PRF whose key is part of the seed, and rely on the fact that the source does not see

the seed and therefore will not be able to cause a collision. A bigger issue is that the above argument relies on "guessing" and therefore has a super-polynomial security loss. This is fixed by using a series of PEFs with progressively larger input sizes $i = 1, \ldots, i_{max} = \omega(\log \lambda)$ and targeting our guessing strategy for a particular input size $i$ which is just slightly larger than the number of queries $q$ made by the source and therefore ensures we guess correctly with inverse polynomial security. Doing this correctly, requires a more careful construction and proof.

*Construction.* We now describe our construction more formally.

Let $i_{max}, j_{max} = \omega(\lambda)$, and $w = j_{max} i_{max}(i_{max}+1)/2$. For $i \in [i_{max}]$, let $(\mathsf{Gen}^i, f^i)$ be a PEF with input size $i$ and lossiness $\ell$. Let $F : \{0,1\}^\lambda \times \{0,1\}^n \to \{0,1\}^v$, $G : \{0,1\}^v \to \{0,1\}^w$. Let $\mathsf{2Ext} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$ be a strong 2-source extractor. We construct an ED-extractor $(\mathsf{SeedGen}, \mathsf{EDExt})$ as follows:

- $\mathsf{SeedGen}(1^\lambda)$: For $i \in [i_{max}], j \in [j_{max}]$, set $k_{i,j} \leftarrow \mathsf{Gen}^i(1^\lambda)$, $K \leftarrow \{0,1\}^\lambda$. Output $\mathsf{seed} = (\{k_{i,j}\}, K)$.
- $\mathsf{EDExt}(s, \mathsf{seed})$: Compute $z = F_K(s)$. Parse $G(z) = \{x_{i,j}\}_{i \in [i_{max}], j \in [j_{max}}$ where $x_{i,j} \in \{0,1\}^i$ for all $i \in [i_{max}], j \in [j_{max}]$. Compute, for all $i \in [i_{max}], j \in [j_{max}]$, $y_{i,j} = f^i_{k_{i,j}}(x_{i,j})$, and set $y = \bigoplus_{i,j} y_{i,j}$. Output $\mathsf{2Ext}(x, y)$.

**Theorem 14.** *Let $e_1 = \alpha - v - 1$, $e_2 = \ell - v - 1$. Assume $\mathsf{2Ext}$ is a $(e_1, e_2, \mathrm{negl}(\lambda))$-extractor, that $F$ is a PRF, $G$ is a PRG, and the PEFs $(\mathsf{Gen}^i, f^i)$ satisfy $\ell$-lossiness. Then $(\mathsf{SeedGen}, \mathsf{EDExt})$ is an $\alpha$-entropy secure ED-extractor.*

*Proof.* For completeness, we describe the hybrid games which we directly adapt from [DVW20], and highlight any differences that occur.

Let $\mathcal{D}, \mathcal{S}$ be a source/distinguisher pair. Let $q$ be the number of extractor queries made by $\mathcal{D}$, and let $i^* = \lceil \log q \rceil + 1$. We consider the following sequence of hybrids.

*Hybrid $H_0$.* This is the ED-Extractor game with a source/distinguisher $\mathcal{D}, \mathcal{S}$. The game proceeds as follows:

- Sample a random bit $b \leftarrow \{0,1\}$ and a random $\mathsf{seed} \leftarrow \mathsf{SeedGen}(1^\lambda)$.
- Run $(s, \mathsf{aux}) \leftarrow \mathcal{D}^{\mathsf{EDExt}(\cdot, \mathsf{seed})}(1^\lambda)$.
- If $b = 0$ set $r = \mathsf{EDExt}(s, \mathsf{seed})$ else if $b = 1$ set $r \leftarrow \{0,1\}^m$.
- Let $b' = \mathcal{S}(1^\lambda, \mathsf{seed}, \mathsf{aux}, r)$.

*Hybrid $H_1$.* We change the way $b'$ is computed. Let $x^*_{i,j} \in \{0,1\}$ denote the values internally computed by the call $\mathsf{EDExt}(s, \mathsf{seed})$ in the experiment. Let $\mathsf{BAD}$ denote the event that, any oracle call from $\mathcal{S}$ to $\mathsf{EDExt}(\cdot, \mathsf{seed})$ sampled the internal value $x_{i^*,j} = x^*_{i^*,j}$ for *all* $j \in [j_{max}]$.

We now set $b' \leftarrow \{0,1\}$ if $\mathsf{BAD}$ occurs (and no modification otherwise).

Hybrids 0 and 1 are indistinguishable. This is because the probability that $\mathsf{BAD}$ occurs is negligible, by PRF security of $F$ and PRG security of $G$, and where we crucially use $2^{i^*} \geq 2q$.

*Hybrid* $H_2$. We now sample $j^* \leftarrow [j_{max}]$ and $x^* \leftarrow \{0,1\}^{i^*}$ in the beginning of the experiment, where we recall $i^* = i^* = \lceil \log q \rceil + 1$. Let GUESS denote the event that (1) BAD does not occur and (2) $j^*$ is the least element in $[j_{max}]$ such that no oracle call from $\mathcal{S}$ to $\mathsf{EDExt}(\cdot, \mathsf{seed})$ sampled the internal value $x_{i^*,j^*} = x^*_{i^*,j^*}$ (which is well-defined if BAD does not occur) and (3) $x^* = x_{i^*,j^*}$.

We now set $b' \leftarrow \{0,1\}$ if $\neg$GUESS occurs (and no modification otherwise).

GUESS occurs with probability at least $\frac{1}{2 \cdot q \cdot j_{max}} - \mathrm{negl}(\lambda)$, and thus any distinguisher in $H_1$ with advantage Adv induces a distinguisher in $H_2$ with advantage $\mathsf{Adv}/p - \mathrm{negl}(\lambda)$, where $p = 2 \cdot q \cdot j_{max}$. Note that we crucially use the fact that GUESS occurring is independent of the view of the adversary.

*Hybrid* $H_3$. We switch how the seed is generated. After sampling $j^* \leftarrow [j_{max}]$ and setting $i^* = \lceil \log q \rceil + 1$ and sampling $x^* \leftarrow \{0,1\}^{i^*}$, we now generate $k_{i^*,j^*} \leftarrow \mathsf{LossyGen}(1^\lambda, x^*)$.

Hybrids 2 and 3 are indistinguishable by security of the PEF.

*Hybrid* $H_4$. We now sample $r \leftarrow \{0,1\}^m$ regardless of $b$.

This is indistinguishable by security of 2Ext. The proof is almost identical to the one of [DVW20, Claims 5.11.1 and 5.11.2]. The only difference is how we argue that $Y_{i^*,j^*} = f_k(x^*)$ has min-entropy conditioned on $L = \{f_k(x)\}_{x \neq x^*}$, $Z = F_K(s)$ the seed of the PRG $G$, and $E$ which denotes whether $\neg$BAD and GUESS hold simultaneously.

We have:

$$\begin{aligned} H_\infty(Y_{i^*,j^*} \mid L, Z, E) &\geq H_\infty(Y_{i^*,j^*} \mid L) - v - 1 \\ &\geq \ell - v - 1, \end{aligned}$$

where the first inequality follows from the fact that $Z$ and $E$ have size $v$ and 1 respectively, and the second from $\ell$-lossiness of the PEF.

*Wrapping up.* We show how to setup the parameters to obtain Theorem 12. We can set $\alpha = \lambda^{\Omega(1)}$, $n = \max(\alpha, \lambda)$, $i_{max} = j_{max} = \lambda^{.1}$, $v = \min\{\alpha/2, \lambda^{.1}\}$, and use PEFs with input size $i$, where $i \in [i_{max}]$, and lossiness and output lengths $k = \ell = n$ (Theorem 11).

This makes use of the Raz 2-source extractor with $e_1 = \alpha - v - 1 \geq \alpha/2 - 1 = \lambda^{\Omega(1)}$ and $e_2 = \ell - v - 1 = (1 - o(n)) \cdot n$, $\delta = 2^{-\lambda^{\Omega(1)}}$ and output size $m = \lambda^{\Omega(1)}$ (Theorem 13). This overall proves Theorem 12.

### A.3 Deterministic Leakage-Resilient MACs

We note here, as already observed by [BHK11], that PEFs with super-logarithmic lossiness $\ell$ directly give deterministic, leakage-resilient MACs with selective unforgeability security. Leakage resilience here denotes the fact that the adversary can learn any efficiently computable function of the secret key, as long as the output size of the function is bounded by some leakage-size parameter $L$. Selective unforgeability security means that the adversary chooses the message $m^*$ on

which it will produce a forgery ahead of time, before seeing the leakage or seeing any authentication tags for chosen messages. We refer the reader to [HLWW13] for a formal definition of leakage-resilient MACs.

**Theorem 15 (Deterministic Leakage-Resilient MACs from OWFs).** *Let $m = \text{poly}(\lambda)$ be any message length and $t \geq \omega(\log \lambda)$ be any tag length. Assuming one-way functions, there exists, a deterministic MAC with message length $m$ and tag length $t$ that satisfies selective unforgeability even given leakage of size $L$, as long as $t - L = \omega(\log \lambda)$. The key length of the MAC is $t \cdot \lambda$.*

Note that the amount of tolerated leakage in the above theorem is optimal relative to the tag size: if $t - L = O(\log \lambda)$ then the adversary can just leak the firs $L$ bits of the tag for some message $m^*$ and guess the remaining $t - L$ bits with inverse polynomial probability. On the other hand, the tolerated leakage is not optimal relative to the key size since the ratio of leakage $L$ to key size is $\leq 1/\lambda$. Under the DDH assumption, [BHK11] shows how to get leakage to key size ratio of $(1 - o(1))$.

Note that, by using complexity leveraging, one can obtain a fully-secure, leakage-resilient MAC with the same parameters, if we suppose the underlying PEF is sub-exponentially secure. This, in turn, can be based on the existence of sub-exponentially secure one-way functions.

We can build the MAC of Theorem 15 from any PEF, by viewing the PEF key $k$ as the MAC key (generated using $\mathsf{Gen}$ for the actual scheme) and $f_k(x)$ as the MAC of message $x$. More formally, if $(\mathsf{Gen}, \mathsf{LossyGen}, f)$ is a PEF, we define the MAC as follows:

- $\mathsf{KeyGen}(1^\lambda)$: Output $k \leftarrow \mathsf{Gen}(1^\lambda)$;
- $\mathsf{Sign}(k, m)$: Output $f_k(m)$;
- $\mathsf{Verify}(k, t, m)$: Output 1 if $t = f_k(m)$, and 0 otherwise.

For security, if $k$ were generated as $k \leftarrow \mathsf{LossyGen}(x^*)$, then $f_k(x^*)$ still would have (at least) $\ell - L$ bits of (min-)entropy given the MAC of *all* other messages $x \neq x^*$ and any leakage on $k$ of size $L$, and would in particular be unpredictable whenever $\ell - L \geq \omega(\log \lambda)$. Moreover such a $k$ is indistinguishable from the MAC key by indistinguishability of the PEF. Note here that we crucially need the reduction to know the target message $x^*$ during the key generation, so that the above only yields a *selectively secure* MAC, where the adversary declares ahead of time the message over which he wants to produce a forgery. One can generically achieve full security using complexity leveraging, thus relying on stronger sub-exponential security of the PEF (which follows by sub-exponentially secure PRGs).

Theorem 15 follows by combining the construction above with Theorem 11.

### A.4 Leakage-Resilient Symmetric Encryption

Next, we remark that any PEF also yields a leakage-resilient symmetric encryption scheme. In a leakage-resilient scheme, the adversary can get some leakage of

the secret key, potentially after making some CPA-encryption queries but before seeing the challenge ciphertext. As in the case of MACs, we assume the leakage consists of an arbitrary efficiently computable function applied to the secret key, as long as the outut size of the function is at most $L$ bits for some leakage parameter $L$. Our scheme is public-coin, meaning that all the internal randomness of the encryption procedure is explicitly contained in the ciphertext. We refer the reader to [HLWW13] for a formal definition of leakage-resilient symmetric-key encryption.

**Theorem 16 (Leakage-Resilient Symmetric Encryption from OWFs).** *For any polynomial leakage parameter $L = L(\lambda)$ and message length $m = m(\lambda)$, assuming one-way functions exists, there exists a public-coin, leakage-resilient symmetric encryption scheme with message length $m$, ciphertext size $m + O(\lambda)$, and key size $O((L + \lambda)\lambda)$.*

We recall the definition of a (strong) seeded extractor.

**Definition 9 (Strong Seeded Extractors).** *An efficient function $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a strong $(k, \epsilon)$-extractor if for every $(n, k)$-source $X$,*

$$\mathsf{SD}((U_d, \mathsf{Ext}(X, U_d)), (U_d, U_m)) \leq \epsilon.$$

**Theorem 17 ( [GUV07]).** *For every constant $\alpha > 0$, and all positive integers $n, k$ and all $\epsilon > 0$, there is an explicit construction of a $(k, \epsilon)$-extractor with input length $n$, output length $m = (1 - \alpha)k - O(\log n + \log(1/\varepsilon))$ and seed length $d = O(\log n + \log(1/\varepsilon))$.*

*Construction.* Fix a polynomial $L = L(\lambda)$ for the amount of tolerated leakage. Let $\ell = L + 3\lambda$ and let $(\mathsf{Gen}, \mathsf{LossyGen}, f)$ be a PEF with input length $\lambda$, lossyness parameter $\ell$, output size $\ell$ and key length $L \cdot \lambda$. Let $\mathsf{Ext}$ be a $(2\lambda, \mathrm{negl}(\lambda))$-extractor with input length $\ell$ output length $\lambda$ as guaranteed by Theorem 17, and let $G : \{0,1\}^\lambda \to \{0,1\}^m$ be a PRG. The encryption scheme is defined as:

- $\mathsf{KeyGen}(1^\lambda)$: Output $k \leftarrow \mathsf{Gen}(1^\lambda)$,
- $\mathsf{Enc}(k, \mu)$: Sample a uniform $\mathsf{seed}$ for $\mathsf{Ext}$, a uniform PEF input $x \leftarrow \{0,1\}^\lambda$ and output
$$\mathsf{ct} = (\mathsf{seed}, x, G(\mathsf{Ext}(f_k(x); \mathsf{seed})) \oplus \mu),$$
- $\mathsf{Dec}(k, \mathsf{ct} = (\mathsf{seed}, x, y))$: Output $\mu = G(\mathsf{Ext}(f_k(x); \mathsf{seed})) \oplus y$.

**Claim 6.** *Assuming the security of the underlying building blocks, the above construction gives a public-coin, symmetric encryption scheme with message length $m$ and leakage bound $L$.*

*Proof.* We switch to an indistinguishable hybrid where we select the value $x^*$ for the challenge ciphertext at the very beginning and set $k \leftarrow \mathsf{LossyGen}(x^*)$. This is indistinguishable even given $k$ in full and therefore certainly given leakage on $k$. We also choose the values $x$ in all other encryption queries uniformly at random from $\{0,1\}^n \setminus \{x^*\}$, which is statistically indistinguishable. Now we

argue that $f_k(x^*)$ has (at least) $\ell - L$ bits of (min-)entropy conditioned on the otuputs of all the encryption queries and any leakage on $k$ of size $L$. As a result, $\mathsf{Ext}(\mathsf{seed}, f_k(x))$ is statistically close to uniformly random. Therefore, we can rely on PRG security to switch $G(\mathsf{Ext}(\mathsf{seed}, f_k(x^*)))$ in hte challenge ciphertext to a uniformly random value. In this case the adversary does not learn anything about $\mu$ which concludes the proof.

Combining the above with Theorem 11 and Theorem 17 gives the parameters of Theorem 16.

### A.5 Symmetric-Key Encryption Secure against Selective Opening Attacks

The goal of this section is to prove the following:

**Theorem 18 (Selective Opening Security from OWFs).** *Assuming one-way functions exist, there exists a symmetric-key encryption scheme that achieves simulation-security against selective opening of keys and randomness (Definition 10).*

In Appendix A.5, we recall the notion of security against selective opening attacks, and provide a simulation-security definition in the context of symmetric-key encryption (Definition 10). In Appendix A.5, we consider a strengthened version of PEFs featuring some equivocability, and provide a construction based on one-way functions (Theorem 18).

**Selective Opening Security.** We first define selective opening security in the symmetric-key setting. In the public-key encryption setting, an adversary, given ciphertexts encrypting correlated messages under different public keys, can adaptively request so-called *openings* of a subset of these ciphertexts. This allows the adversary to obtain either the randomness used to encrypt the subset of ciphertexts (sometimes refered to as sender corruptions) or subset of secret keys allowing to decrypt (sometimes refered to as receiver corruptions).

In the symmetric-key setting, an adversary receives ciphertexts of correlated messages encrypted under different secret keys. We focus on the setting where openings correspond to secret keys, which allow the adversary to decrypt the associated subset of ciphertexts. In fact, we will also handle randomness opening for free, as our scheme is public-coin, namely, all the randomness of the encryption is included as part of the ciphertext. In other words, we are able to provide security against both kinds of openings.

In terms of definitions, there are mainly two ways of formalizing selective opening security, either through a weaker indistinguishability-based definition, or a stronger simulation-based definition. The main drawback of the indistinguishability-based definition is that it significantly restricts the possible correlations of the encrypted messages. Fortunately, we manage to construct the stronger notion of

simulation-based selective opening security against openings of both keys and randomness, which the notion we focus on in this paper.

One drawback of our simulation-based definition is that we only allow a single opening query per secret key in the experiment. This is essentially inherent: a secret key $\mathsf{sk}$ cannot explain more than $\mathsf{sk}$ ciphertexts to any arbitrary messages, or would otherwise violate entropy lower bounds.

**Definition 10 (Selective Opening Simulation Security in the Symmetric-Key Setting).** *Let $d = d(\lambda)$ be a polynomial. Let $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a symmetric-key encryption scheme. For PPT algorithms $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ and $\mathcal{D} = (\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3)$ and $\mathcal{P}$, we consider the following experiments:*

***Experiment*** $\mathsf{Exp}^{\mathsf{real}}_{\mathsf{Sim-SOA}}(\mathcal{A}, 1^\lambda)$*:*

1. *Sample, for all $i \in [d]$, $\mathsf{sk}_i \leftarrow \mathsf{KeyGen}(1^\lambda)$.*
2. *The adversary computes $(\mathcal{D}, \mathsf{state}_1) \leftarrow \mathcal{A}_1^{\mathsf{Enc}(\mathsf{sk}_1,\cdot),\ldots,\mathsf{Enc}(\mathsf{sk}_d,\cdot)}(1^\lambda)$, where $\mathcal{D}$ is a distribution over $\mathcal{M}^n$, where $\mathcal{M}$ is the message space of the encryption scheme.*
3. *Sample $\overrightarrow{m}^* \leftarrow \mathcal{D}$.*
4. *Compute for all $i \in [d]$: $\mathsf{ct}_i^* = \mathsf{Enc}(\mathsf{sk}_i, m_i^*; r_i^*)$ where $r_i^*$ are the random coins used to encrypt.*
5. *The adversary computes $(I, \mathsf{state}_2) \leftarrow \mathcal{A}_2^{\mathsf{Enc}(\mathsf{sk}_1,\cdot),\ldots,\mathsf{Enc}(\mathsf{sk}_d,\cdot)}(\mathsf{state}_1, \mathsf{ct}_1^*, \cdots, \mathsf{ct}_d^*)$, where $I \subseteq [d]$.*
6. *The adversary computes $\mathsf{output} \leftarrow \mathcal{A}_3^{\mathsf{Enc}(\mathsf{sk}_1,\cdot),\ldots,\mathsf{Enc}(\mathsf{sk}_d,\cdot)}(\mathsf{state}_2, \{m_i^*, r_i^*, \mathsf{sk}_i\}_{i\in I})$.*
7. *The output of the experiment is the bit $\mathcal{P}(\mathcal{D}, \overrightarrow{m}^*, I, \mathsf{output})$.*

***Experiment*** $\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{Sim-SOA}}(\mathcal{D}, 1^\lambda)$*:*

1. *The simulator samples $(\mathcal{D}, \mathsf{state}_1) \leftarrow \mathcal{D}_1(1^\lambda)$.*
2. *Sample $\overrightarrow{m}^* \leftarrow \mathcal{D}$.*
3. *The simulator computes $(I, \mathsf{state}_2) \leftarrow \mathcal{D}_2(\mathsf{state}_1, 1^{|m_1^*|}, \ldots, 1^{|m_d^*|})$.*
4. *The simulator computes $\mathsf{output} \leftarrow \mathcal{D}_3(\mathsf{state}_2, \{m_i^*\}_{i\in I})$.*
5. *The output of the experiment is the bit $\mathcal{P}(\mathcal{D}, \overrightarrow{m}^*, I, \mathsf{output})$.*

*We say that $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is $\mathsf{Sim-SOA}$-secure if for all polynomial $d = d(\lambda)$, for all PPT adversary $\mathcal{A}$ in $\mathsf{Exp}^{\mathsf{real}}_{\mathsf{Sim-SOA}}(\mathcal{A}, 1^\lambda)$ and for all distinguisher $\mathcal{P}$, there exists a simulator $\mathcal{D}$ in $\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{Sim-SOA}}(\mathcal{D}, 1^\lambda)$ such that:*

$$\left|\Pr[\mathsf{Exp}^{\mathsf{real}}_{\mathsf{Sim-SOA}}(\mathcal{A}, 1^\lambda) = 1] \;\; \Pr[\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{Sim-SOA}}(\mathcal{D}, 1^\lambda) = 1]\right| \leq \mathrm{negl}(\lambda).$$

**Equivocal T-ALBOs and PEFs** Our main building block to achieve selective opening security is a stronger notion of PEFs that we call *equivocal* PEFs.

**Definition 11 (Equivocable Pseudo-Entropy Functions).** *The syntax of an equivocal PEF with input size $n$ and output size $m$ is similar to the one of a PEF (Definition 6), except that $\mathsf{LossyGen}(1^\lambda, x^*)$ outputs $(k, \mathsf{state})$ where $\mathsf{state}$ is some secret state. We require an additional PPT algorithm*

– $\mathsf{Equivocate}(\mathsf{state}, y) \to k^*_{sim}$*: on input* $y \in \{0,1\}^m$ *(in the output space of the PEF) and* state*, output* $k^*_{sim}$ *(in the key of the PEF).*

*We require the following properties:*

– *Consistency: For any* $x^* \in \{0,1\}^t, y \in \{0,1\}^m$*: if* $(k^*, \mathsf{state}) \leftarrow \mathsf{LossyGen}(1^\lambda, \mathsf{tag}^*, x^*)$ *and* $k^*_{sim} \leftarrow \mathsf{Equivocate}(\mathsf{state}, y)$ *then* $f_k(x) = f_{k^*_{sim}}(x)$ *for all* $x \neq x^*$*. Furthermore,* $f_{k^*_{sim}}(x^*) = y$*.*
– *Indistinguishability: The following distributions are computationally indistinguishable for all* $x^* \in \{0,1\}^n$*:*

$$(x^*, k) \overset{c}{\approx} (x^*, k^*_{sim}),$$

*where* $x^* \leftarrow \{0,1\}^n$*,* $k \leftarrow \mathsf{Gen}(1^\lambda)$*,* $(k^*, \mathsf{state}) \leftarrow \mathsf{LossyGen}(1^\lambda, \mathsf{tag}^*, x^*)$*,* $y \leftarrow \{0,1\}^m$*, and* $k^*_{sim} \leftarrow \mathsf{Equivocate}(\mathsf{state}, y)$*.*

*Remark 2 (Relation with Somewhere Equivocable PRFs ( [HJO+16])).* We note, perhaps surprisingly, that the notion of equivocable PEFs exactly matches the notion of (1-point equivocable) *somewhere equivocable PRFs* (SEPRFs) [HJO+16]. In fact, [HJO+16] builds SEPRFs from one-way functions using techniques very similar to the distributed point function of [BGI15], which, as previously discussed, are connected to our notion of T-A**L**BO and therefore PEFs. Interestingly, [HJO+16] introduced SEPRFs in a yet different context from our other applications (but not entirely unrelated to selective opening security), that is, adaptive security of Yao's garbled circuits. Indeed, it was used to construct a form of (somewhere) non-committing symmetric-key encryption, and non-committing public-key encryption is known to imply selective key-opening security in the public-key setting [HPW15].

Next, we show the following:

**Theorem 19 (Equivocal PEFs from OWFs).** *Let* $\ell = \ell(\lambda)$ *and* $t = t(\lambda)$ *be any polynomials, and let* $m \leq 3\lambda + t$*. Assuming the existence of one-way functions, there exists an equivocal PEF with input length* $t$*, output length* $\ell$ *and lossiness* $\ell$*.*

*Construction.* We build an equivocal PEF by building an associate form of *equivocal T-A**L**BO*, and seeing the resulting T-A**L**BO as a PEF (as in Claim 5).

**Definition 12 (Equivocal T-ALBO).** *The syntax of an equivocal T-A**L**BO is similar to the one of a T-A**L**BO (Definition 2), except that* $\mathsf{LossyGen}(1^\lambda, \mathsf{tag}^*, x^*)$ *outputs* $(\mathsf{fk}, \mathsf{state})$ *where* state *is some secret state. We require an additional PPT algorithm*

– $\mathsf{Equivocate}(\mathsf{state}, y) \to x^*_{sim}$*: on input* $y \in \{0,1\}^m$ *(that is, the output space of the T-A**L**BO) and* state*, output* $x^*_{sim} \in \{0,1\}^n$ *(that is, the input space of the T-A**L**BO).*

*We require the following properties:*

- *Consistency: For any* $x^* \in \{0,1\}^n, \mathsf{tag}^* \in \{0,1\}^t, y \in \{0,1\}^m$*: if* $(\mathsf{fk}, \mathsf{state}) \leftarrow \mathsf{LossyGen}(1^\lambda, \mathsf{tag}^*, x^*)$ *and* $x^*_{sim} \leftarrow \mathsf{Equivocate}(\mathsf{state}, y)$ *then* $F_{\mathsf{fk},\mathsf{tag}}(x^*) = F_{\mathsf{fk},\mathsf{tag}}(x^*_{sim})$ *for all* $\mathsf{tag} \neq \mathsf{tag}^*$*. Furthermore,* $F_{\mathsf{fk},\mathsf{tag}^*}(x^*_{sim}) = y$*.*
- *Indistinguishability: The following distributions are computationally indistinguishable for all* $\mathsf{tag}^* \in \{0,1\}^t$*:*

$$(\mathsf{tag}^*, \mathsf{fk}_{inj}, x^*) \overset{\mathrm{c}}{\approx} (\mathsf{tag}^*, \mathsf{fk}_{los}, x^*_{sim}),$$

*where* $x^* \leftarrow \{0,1\}^n$*,* $\mathsf{fk}_{inj} \leftarrow \mathsf{InjectiveGen}(1^\lambda)$*,* $(\mathsf{fk}_{los}, \mathsf{state}) \leftarrow \mathsf{LossyGen}(1^\lambda, \mathsf{tag}^*, x^*)$*,* $y \leftarrow \{0,1\}^m$*, and* $x^*_{sim} \leftarrow \mathsf{Equivocate}(\mathsf{state}, y)$*.*

Our construction of an equivocal T-A**L**BO follows directly from our construction of entropy-preserving T-A**L**BO (Theorem 4). We only slightly modify the key generation in lossy mode to output a state, and define our new equivocation algorithm:

- $\widetilde{\mathsf{LossyGen}}(1^\lambda, \mathsf{tag}^*, x^*)$: Sample $\mathsf{fk} \leftarrow \overline{\mathsf{LossyGen}}(1^\lambda, \mathsf{tag}^*, x^*)$, which also internally samples $x^*_1 \leftarrow \{0,1\}^\lambda$. If $F_{\mathsf{fk},\mathsf{tag}^*}(x^*) = F_{\mathsf{fk},\mathsf{tag}^*}(x^*_1)$, repeat the above. Otherwise, compute $y^* = \overline{F}_{\mathsf{fk},\mathsf{tag}^*}(x^*)$ and $y_1 = \overline{F}_{\mathsf{fk},\mathsf{tag}^*}(x^*_1)$. Set $a = e \cdot (y^* - y_1)^{-1}$ and $\mathsf{state} = (\mathsf{tag}^*, x^*, x^*_1)$. Output $(\mathsf{fk}, a, \mathsf{state})$.
- $\mathsf{Equivocate}(\mathsf{state}, y)$: On input $\mathsf{state} = (\mathsf{tag}^*, x^*, x^*_1)$ and $y \in \{0,1\}$, compute $b_0 = \widetilde{F}_{\mathsf{fk},\mathsf{tag}^*}(x^*)$, $b_1 = \widetilde{F}_{\mathsf{fk},\mathsf{tag}^*}(x^*_1)$. If $b_0 = y$, output $x^*$, and output $x^*_1$ otherwise.

We keep the algorithms $\widetilde{\mathsf{InjectiveGen}}$ and $\widetilde{F}$ unchanged, as in the construction of Claim 3. Note that $\mathsf{Equivocate}$ is well-defined as $b_0 \neq b_1$ by construction, so that either $y = b_0$ or $y = b_1$.

Consistency follows by the proof of lossiness of Claim 3, where we proved that $F_{\mathsf{fk},\mathsf{tag}}(x^*) = F_{\mathsf{fk},\mathsf{tag}}(x^*_1)$ for all $\mathsf{tag} \neq \mathsf{tag}^*$. The equality $F_{\mathsf{fk},\mathsf{tag}^*}(x^*_{sim}) = y$ is by construction.

For indistinguishability, we have by indistinguishability of the entropy-preserving T-A**L**BO that for all $\mathsf{tag}^* \in \{0,1\}^t$, $(\mathsf{tag}^*, \mathsf{fk}_{inj}, x^*) \overset{\mathrm{c}}{\approx} (\mathsf{tag}^*, \mathsf{fk}_{los}, x^*)$ where $x^* \leftarrow \{0,1\}^n$, $\mathsf{fk}_{inj} \leftarrow \mathsf{InjectiveGen}(1^\lambda)$, and $\mathsf{fk}_{los} \leftarrow \mathsf{LossyGen}(1^\lambda, \mathsf{tag}^*, x^*)$. Then, we showed in the proof of Claim 3 that the views $(\mathsf{tag}^*, \mathsf{fk}_{los}, x^*)$ and $(\mathsf{tag}^*, \mathsf{fk}_{los}, x^*_1)$ are identically distributed. Therefore, the distribution $(\mathsf{tag}^*, \mathsf{fk}, x^*_{sim})$, where $x^* \leftarrow \{0,1\}^n, (\mathsf{fk}, \mathsf{state}) \leftarrow \mathsf{LossyGen}(1^\lambda, \mathsf{tag}^*, x^*), y \leftarrow \{0,1\}^m, x^*_{sim} \leftarrow \mathsf{Equivocate}(\mathsf{state}, y)$, which picks a random one out of the two, is also indistinguishable from $(\mathsf{tag}^*, \mathsf{fk}_{inj}, x^*)$.

Again, output size can be amplified by concatenating the output of many such equivocal T-A**L**BOs with bit outputs. Combined with the above, this gives the following:

**Theorem 20.** *Let* $k = k(\lambda)$ *and* $t = t(\lambda)$ *be any polynomials. Assuming the existence of one-way functions, there exists an equivocal T-A**L**BO with input size* $\{0,1\}^{k\lambda}$*, tag space* $\{0,1\}^t$*, and output space* $\{0,1\}^k$*.*

As in Claim 5, equivocable PEFs can be readily constructed from any equivocable T-A**L**BO by defining inputs of the PEF as T-A**L**BO tags, and PEF keys as T-A**L**BO inputs. This yields Theorem 19.

**Selective Opening Security from Equivocal PEFs** We now describe our construction of a symmetric-key encryption scheme which is secure against selective opening of keys and randomness, thus proving Theorem 18.

Our main idea will be to use an equivocal PEF to construct a symmetric-key encryption scheme which enjoys a (weak) key-equivocation property. Achieving security against selective opening of randomness will be for free as our scheme is public-coin.

*Construction.* Let $(\mathsf{Gen}, \mathsf{LossyGen}, f)$ be an equivocable PEF (Definition 11) with input size $n$ and output size $m$. We define the following encryption scheme:

- $\mathsf{KeyGen}(1^\lambda)$: Sample $k \leftarrow \mathsf{Gen}(1^\lambda)$, and output $\mathsf{sk} = k$.
- $\mathsf{Enc}(\mathsf{sk}, m)$: Sample $x \leftarrow \{0,1\}^n$ and output
$$\mathsf{ct} = (x, f_k(x) \oplus m).$$
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: Parse $\mathsf{ct}$ as $\mathsf{ct} = (x, y)$, compute
$$m = y \oplus f_k(x).$$

**Theorem 21.** *Suppose* $(\mathsf{Gen}, \mathsf{LossyGen}, f)$ *is an equivocable PEF (Definition 11). Then* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is simulation-secure against selective openings of keys and randomness (Definition 10).*

*Proof.* Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ be any PPT adversary in $\mathsf{Exp}^{\mathsf{real}}_{\mathsf{Sim-SOA}}$, and $\mathcal{P}$ be a distinguisher. We define our simulator $\mathcal{D} = (\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3)$ for $\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{Sim-SOA}}$ as follows.

- $\mathcal{D}_1(1^\lambda)$: For all $i \in [d]$, sample $x_i^* \leftarrow \{0,1\}^n$, and compute $(k_i^*, \mathsf{state}_i^{PEF}) \leftarrow \mathsf{LossyGen}(1^\lambda, x^*)$. Compute $(\mathcal{D}, \mathsf{state}_1^{\mathcal{A}}) \leftarrow \mathcal{A}_1^{\mathsf{Enc}(\mathsf{sk}_1,\cdot),\ldots,\mathsf{Enc}(\mathsf{sk}_d,\cdot)}(1^\lambda)$ where encryptions queries as answered using $k_i^*$, as:
$$\mathsf{Enc}(\mathsf{sk}_i, m) = (x, f_{k_i^*}(x) \oplus m),$$
where $x \leftarrow \{0,1\}^n$. Set $\mathsf{state}_1 = (\mathsf{state}_1^{\mathcal{A}}, \{\mathsf{state}_i^{PEF}, x_i^*\}_{i\in[d]})$. Output $(\mathcal{S}, \mathsf{state}_1)$.
- $\mathcal{D}_2(\mathsf{state}_1, 1^{|m_1|}, \ldots, 1^{|m_d|})$: Compute, for all $i \in [d]$:
$$\mathsf{ct}_i^* = (x_i^*, y_i^*),$$
where $y_i^* \leftarrow \{0,1\}^m$.
Compute $(I, \mathsf{state}_2^{\mathcal{A}}) \leftarrow \mathcal{A}_2^{\mathsf{Enc}(\mathsf{sk}_1,\cdot),\ldots,\mathsf{Enc}(\mathsf{sk}_d,\cdot)}(\mathsf{state}_1^{\mathcal{A}}, \mathsf{ct}_1^*, \cdots, \mathsf{ct}_d^*)$ where again encryption queries are answered using $k_i^*$, as:
$$\mathsf{Enc}(\mathsf{sk}_i, m) = (x, f_{k_i^*}(x) \oplus m).$$
Output $(I, \mathsf{state}_2 = (\mathsf{state}_2^{\mathcal{A}}, \mathsf{state}_1))$.
- $\mathcal{D}_3(\mathsf{state}_2, \{m_i^*\}_{i\in I})$ : Compute, for all $i \in I$, $k_{i,sim}^* \leftarrow \mathsf{Equivocate}(\mathsf{state}_i^{PEF}, m_i^*)$. Compute and output $\mathsf{output} \leftarrow \mathcal{A}_3^{\mathsf{Enc}(\mathsf{sk}_1,\cdot),\ldots,\mathsf{Enc}(\mathsf{sk}_d,\cdot)}(\mathsf{state}_2, \{m_i^*, x_i^*, k_{i,sim}^*\}_{i\in I})$.

We show that the distribution of $\mathsf{Exp}^{\mathsf{real}}_{\mathsf{Sim-SOA}}(\mathcal{A}, 1^\lambda)$ and $\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{Sim-SOA}}(\mathcal{D}, 1^\lambda)$ are computationally indistinguishable by considering a series of hybrid distributions.

*Hybrid* $H_0$. This is the distribution induced by $\mathsf{Exp}^{\mathsf{real}}_{\mathsf{Sim-SOA}}(\mathcal{A}, 1^\lambda)$.

*Hybrid* $H_1$. We make the following changes:

- We change the way the randomness used to compute the challenge ciphertexts is sampled. We now sample, for all $i \in [d]$, $x_i^* \leftarrow \{0,1\}^n$ before sampling secret keys.
- We change how we answer encryption queries. We abort the experiment if any encryption query for secret key $i$ samples $x_i^*$ as randomness.

*Hybrid* $H_{2,i}$, $i \in [d]$. We make the following changes:

- We change the way the setup of the experiment is performed. We now sample $x_i^* \leftarrow \{0,1\}^n$ and set $(k_i^*, \mathsf{state}_i^{PEF}) \leftarrow \mathsf{LossyGen}(1^\lambda, x_i^*)$. All the encryption queries throughout the experiment are now answered using $k_i^*$.
- We change the way the challenge ciphertext for index $i$ is computed. We now compute
$$\mathsf{ct}_i^* = (x^*, y_i^* \oplus m_i^*),$$
where $y_i^* \leftarrow \{0,1\}^m$.
- We change the way the values $r_i^*, \mathsf{sk}_i$ given to $\mathcal{A}_3$ are computed. Given $(m_i^*)_{i \in I}$, we now compute, if $i \in I$:
$$k_{i,sim}^* \leftarrow \mathsf{Equivocate}(\mathsf{state}_i^{PEF}, y_i^*),$$
and $\mathcal{A}_3$ is now given as input $(x_i^*, k_{i,sim}^*)$.

*Hybrid* $H_3$. We make the following changes:

- We do not abort anymore if an encryption query for secret key $i$ samples $x = x_i^*$.
- We change again the way the challenge ciphertexts are computed. We now compute for all $i \in [d]$
$$\mathsf{ct}_i^* = (x^*, y_i^*),$$
where $y_i^* \leftarrow \{0,1\}^m$.
- We change the way the values $\mathsf{sk}_i$ given to $\mathcal{A}_3$ are computed. Given $(m_i^*)_{i \in I}$, we now compute for all $i \in I$:
$$k_{i,sim}^* \leftarrow \mathsf{Equivocate}(\mathsf{state}_i^{PEF}, y_i^* \oplus m_i^*).$$

Observe that the view generated by hybrid $H_3$ now corresponds to the view generated by the simulator $\mathcal{D} = (\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3)$ in $\mathsf{Exp}^{\mathsf{ideal}}_{\mathsf{Sim-SOA}}(\mathcal{D}, 1^\lambda)$.

We now prove that the successive hybrids are indistinguishable.

**Claim 7.** *The views generated in hybrids $H_0$ and $H_1$ are within negligible statistical distance.*

*Proof.* The only difference between the two hybrids occurs whenever an encryption query for index $i$ samples as randomness $x = x_i^*$. This happens with probability $1/2^n$, and by union bound over the polynomial number $Q$ of encryption queries made by $\mathcal{A}$ in $\mathsf{Exp}^{\mathsf{real}}_{\mathsf{Sim-SOA}}(\mathcal{A}, 1^\lambda)$, this occurs with probability at most $Q/2^n = \mathrm{negl}(\lambda)$.

**Claim 8.** *Suppose the equivocal PEF* $(\mathsf{Gen}, \mathsf{LossyGen}, f)$ *satisfies consistency and indistinguishability. Then, the views generated in hybrids* $H_1$ *and* $H_{2,1}$ *are indistinguishable. Similarly, the views generated in hybrids* $H_{2,i}$ *and* $H_{2,i+1}$ *are indistinguishable for all* $i \in \{1, \ldots, d-1\}$.

*Proof.* Let us argue that $H_0$ and $H_{1,1}$ are indistinguishable: the proof for hybrids $H_{1,i}$ and $H_{1,i+1}$, $i \in [d]$ is identical.

By indistinguishability of $(\mathsf{Gen}, \mathsf{LossyGen}, f)$, we have:

$$(x_1^*, k_1, f_{k_1}(x_1^*)) \overset{\mathrm{c}}{\approx} (x^*, k_{1,sim}^*, y_1)$$

where $x_1^* \leftarrow \{0,1\}^n$, $k_1 \leftarrow \mathsf{KeyGen}(1^\lambda)$, $(k_1^*, \mathsf{state}_1^{PEF}) \leftarrow \mathsf{LossyGen}(1^\lambda, x_1^*)$ and $k_{1,sim}^* \leftarrow \mathsf{Equivocate}(\mathsf{state}_1^{PEF}, y)$. This is because one can compute the third element of the distributions $(x, k, y)$ as $y = f_k(x)$, by consistency in lossy mode.

It remains to argue that the distribution of the answers to the encryption queries are indistinguishable. This follows by consistency, which ensures that $(x, f_{k_1^*}(x))$ is identically distributed to $(x, f_{k_{1,sim}^*}(x))$ over the randomness of $x \leftarrow \{0,1\}^n$, unless $x = x_1^*$, but both hybrids abort the experiment in that case.

**Claim 9.** *The views generated in hybrids* $H_{1,d}$ *and* $H_2$ *are identically distributed.*

*Proof.* For all $i \in [d]$ and all $m_i^* \in \{0,1\}^m$, the distributions $u_i \oplus m_i^*$ and $u_i^*$ are uniform over $\{0,1\}^m$ over the randomness of $u_i^* \leftarrow \{0,1\}^m$ alone. Setting $y_i^* = u_i \oplus m_i^*$ yields hybrid $H_{2,d}$, and $y_i^* = u_i$ hybrid $H_3$.

Finally, the probability of aborting because an encryption query samples $x = x_i^*$ is $Q/2^n$ (where $Q$ is the number of encryption queries made by $\mathcal{A}$), which is negligible.

Combined with Theorem 19, we obtain Theorem 18.

## B Application of T-AIBOs to CCA Security

We prove in this section the following theorem:

**Theorem 22 (CCA Encryption from Strong Trapdoor Functions).** *Let* $d = d(\lambda)$, $n = n(\lambda) = \omega(\log \lambda)$, $\rho = d \cdot n$, *and* $m = \max(n+1, \lambda)$. *Let* $\mathsf{TDF}$ *be a trapdoor function with input length* $\rho$. *Suppose that no time* $T = 2^n \cdot \mathrm{poly}(\lambda)$ *adversary can invert* $\mathsf{TDF}$ *with probability* $\frac{2^d}{2^\rho} \cdot \epsilon$ *for any non-negligible* $\epsilon$. *Assume furthermore the existence of an injective PRG* $G : \{0,1\}^n \to \{0,1\}^m$.

*Then there exists a CCA-secure (public-key) encryption scheme.*

In Appendix B.1, we recall the notion of encryption schemes with randomness recovery, and show that a standard construction from trapdoor functions satisfies a special form of leakage resilience Lemma 1. In Appendix B.2, we show how to build a public-key CCA-secure encryption scheme assuming the existence of strongly secure trapdoor functions, proving Theorem 22.

### B.1 Encryption with Randomness Recovery

Another core component of our construction is a public-key encryption scheme with randomness recovery, which we base on trapdoor functions (TDF). Informally, such an encryption has two additional properties over standard public-key encryption, namely (1) the decryption algorithm also recovers the random coins used to encrypt, and (2) one can alternatively decrypt ciphertexts using the random coins used to encrypt (as opposed to traditionally with the secret key).

**Definition 13 (Public-Key Encryption with Randomness Recovery).** *A public-key encryption scheme with randomness recovery* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Recover})$ *has the following syntax:*

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$*: On input the security parameter, output a public* $\mathsf{pk}$ *and a secret key* $\mathsf{sk}$*.*
- $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m))$*: On input a public key* $\mathsf{pk}$ *and a message* $m$*, samples some random coins* $r$ *and output a ciphertext* $\mathsf{ct} = \mathsf{Enc}(\mathsf{pk}, m; r)$*.*
- $(m, r) \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$*: On input a secret key* $\mathsf{sk}$ *and a ciphertext* $\mathsf{ct}$*, output a message* $m$ *and random coins* $r$*.*
- $m \leftarrow \mathsf{Recover}(\mathsf{pk}, \mathsf{ct}, r)$*: On input a public key* $\mathsf{pk}$*, a ciphertext* $\mathsf{ct}$ *and random coins* $r$*, output a message* $m$*.*

*We require the following correctness properties:*

*Correctness. We require that correctness holds perfectly, except with negligible probability over* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$*, namely:*

$$\Pr[\exists m, r,\ \mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m; r) \neq (m, r) \vee \mathsf{Recover}(\mathsf{pk}, \mathsf{Enc}(pk, m; r)) \neq m] \leq \mathrm{negl}(\lambda),$$

*over the probability of* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$*.*

*Security. We require standard CPA security.*

For our constructions, we will additionally need some form of leakage resilience, namely that security holds even given some particular leakage on the *random coins* used to encrypt. Note that even though one can generically add leakage resilience against randomness leakage by using strong seeded extractors, this does not preserve randomness recovery.

Instead, we will directly show that the particular construction of public-key encryption with randomness recovery from trapdoor functions of [HKW20] is resilient against a particular form of leakage generated by a T-A**I**BO, assuming strong security of the underlying trapdoor function. We first recall the definition of an injective trapdoor function.

**Definition 14 (Injective Trapdoor Functions (TDF)).** *An injective trapdoor function family* $\mathsf{TDF} = (\mathsf{TDF.Setup}, \mathsf{TDF.Eval}, \mathsf{TDF.Invert})$ *with input length* $\rho = \mathrm{poly}(\lambda)$ *and output length* $m = \mathrm{poly}(\lambda)$ *has the following syntax:*

- $\mathsf{TDF.Setup}(1^\lambda)$*: on input the security parameter, output a public key* $\mathsf{pk}$ *and a secret key* $\mathsf{sk}$*.*
- $\mathsf{TDF.Eval}(\mathsf{pk}, x)$*: a deterministic algorithm, which, on input a public key* $\mathsf{pk}$ *and an input* $x \in \{0,1\}^\rho$*, output* $y \in \{0,1\}^m$*.*
- $\mathsf{TDF.Invert}(\mathsf{sk}, y)$*: on input a secret key* $\mathsf{sk}$ *and an output* $y \in \{0,1\}^m$*, output* $x \in \{0,1\}^\rho$*.*

*We require the following properties:*

*Correctness. We require that:*

$$\Pr[\exists x, \mathsf{TDF.Invert}(\mathsf{sk}, \mathsf{TDF.Eval}(\mathsf{pk}, x)) \neq x] \leq \mathrm{negl}(\lambda),$$

*over the randomness of* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{TDF.Setup}(1^\lambda)$*, namely, that with overwhelming probability over the setup alone, inversion is perfectly correct (which also implies that* $\mathsf{TDF.Eval}(\mathsf{pk}, \cdot)$ *is injective).*

*Security. We say that* $\mathsf{TDF}$ *is hard to invert if for any PPT adversary* $\mathcal{A}$*, there exists a negligible function* $\epsilon$ *such that:*

$$\Pr[x \leftarrow \mathcal{A}(\mathsf{pk}, y)] \leq \epsilon,$$

*over the randomness of* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{TDF.Setup}$*,* $x \leftarrow \{0,1\}^\rho$*, and where* $y = \mathsf{TDF.Eval}(\mathsf{pk}, x)$*.*

*We will also consider strengthened forms of security where* $\mathcal{A}$ *is allowed to run in specific, potentially super-polynomial time* $T$*, and require his success probability to be at most* $\epsilon$ *for some specific function* $\epsilon$*.*

Next, we recall the construction of a randomness-recoverable PKE from any TDF [HKW20].

Let $\mathsf{TDF} = (\mathsf{TDF.Setup}, \mathsf{TDF.Eval}, \mathsf{TDF.Invert})$ be a trapdoor function with input space $\{0,1\}^\rho$. Let $\mathsf{hc}$ denote the Goldreich-Levin hard-core bit [GL89]. We define the following encryption scheme:

- $\mathsf{CPA.KeyGen}(1^\lambda)$: Sample $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{TDF.Setup}$. Sample some randomness $\mathsf{coins}$ for $\mathsf{hc}$, and output $(\mathsf{CPA.pk} = (\mathsf{pk}, \mathsf{coins}), \mathsf{CPA.sk} = (\mathsf{sk}, \mathsf{coins}))$.
- $\mathsf{CPA.Enc}(\mathsf{CPA.pk}, m \in \{0,1\})$: Sample $r \leftarrow \{0,1\}^\rho$, and output

$$\mathsf{ct} = (\mathsf{TDF.Eval}(\mathsf{pk}, r), \mathsf{hc}(r; \mathsf{coins}) \oplus m).$$

- $\mathsf{CPA.Dec}(\mathsf{CPA.sk}, \mathsf{ct})$: On input $\mathsf{ct} = (z, b)$, compute $r = \mathsf{TDF.Invert}(\mathsf{sk}, z)$, and output $m = \mathsf{hc}(r; \mathsf{coins}) \oplus b$.
- $\mathsf{CPA.Recover}(\mathsf{pk}, \mathsf{ct}, r)$: On input $\mathsf{ct} = (z, b)$, compute $z' = \mathsf{TDF.Eval}(\mathsf{pk}, r)$. Abort if $z \neq z'$. Otherwise output $m = \mathsf{hc}(r; \mathsf{coins}) \oplus b$.

Note that the random coins used by $\mathsf{CPA.Enc}$ correspond exactly to a random input to $\mathsf{TDF}$.

Next, we show that this particular encryption scheme satisfies a specific form of leakage-resilience, provided the trapdoor function $\mathsf{TDF}$ is secure enough.

**Lemma 1 (Leakage-Resilience of $\mathsf{CPA}$).** *Let $n = n(\lambda)$, $d = d(\lambda)$ and $t = t(\lambda)$ and $\ell = \ell(\lambda)$ be polynomials. Let $\rho = d{\cdot}n$. Let $\mathsf{TDF} = (\mathsf{TDF.Setup}, \mathsf{TDF.Eval}, \mathsf{TDF.Invert})$ be a trapdoor function with input space $\{0,1\}^\rho$. Let $(\mathsf{InjectiveGen}, \mathsf{LossyGen}, F)$ be the specific T-A**I**BO of Section 4.2 with input space $\{0,1\}^n$ and tag space $\{0,1\}^t$ and lossiness $\ell$.*

*For all $m \in \{0,1\}$ and all $\mathsf{tag}^* \in \{0,1\}^t$, consider the two following distributions:*

$$\big(\{\mathsf{fk}_i\}_{i\in[d]}, \mathsf{pk}, \mathsf{TDF.Eval}_{\mathsf{pk}}(r),\ \{F_{\mathsf{fk}_i,\mathsf{tag}^*}(r_i)\}_{i\in[d]},\ \mathsf{coins}, \mathsf{hc}(r;\mathsf{coins}) \oplus m\big)$$
$$\big(\{\mathsf{fk}_i\}_{i\in[d]}, \mathsf{pk}, \mathsf{TDF.Eval}_{\mathsf{pk}}(r),\ \{F_{\mathsf{fk}_i,\mathsf{tag}^*}(r_i)\}_{i\in[d]},\ \mathsf{coins}, \mathsf{hc}(r;\mathsf{coins})\big),$$

*where $r = (r_1 \| \ldots, \| r_d) \leftarrow \{0,1\}^\rho$, $\mathsf{fk}_i \leftarrow \mathsf{LossyGen}(\mathsf{tag}^*, r_i)$, $\mathsf{pk} \leftarrow \mathsf{TDF.Setup}$, and $\mathsf{hc}$ denotes the Goldreich-Levin hard-core bit [GL89] and $\mathsf{coins}$ uniformly sampled randomness for $\mathsf{hc}$.*

*Suppose that no time $T$ adversary can invert $\mathsf{TDF}$ with probability $\frac{2^{\ell d}}{2^\rho} \cdot \epsilon$ for any non-negligible $\epsilon$, and suppose that that the T-A**I**BO satisfies $\ell$-lossiness. Then no time $T \cdot \mathrm{poly}(\lambda)$ adversary can distinguish the two distributions above with non-negligible success probability.*

*Proof.* Suppose there exists an PPT adversary with runtime $T$ that distinguishes the two distributions above with non-negligible success probability $\epsilon$. Then there exists a predictor $\mathcal{P}$ with runtime $\mathrm{poly}(\rho) \cdot T$, which, on input $\big(\mathsf{TDF.Eval}_{pk}(r),\ \{F_{\mathsf{fk}_i,\mathsf{tag}^*}(r)\}_{i\in[d]}\big)$, outputs $r$ with non-negligible probability $\Omega(\epsilon)$.

Given such a predictor $\mathcal{P}$, we build an inverter for $\mathsf{TDF}$ as follows.

- $\mathcal{I}(\mathsf{TDF.Eval}_{\mathsf{pk}}(r))$: Sample, for $i \in [d]$, $r_i^* \leftarrow \{0,1\}^n$, and set $r^* = \mathcal{P}(\big(\mathsf{TDF.Eval}_{\mathsf{pk}}(r),\ \{F_{\mathsf{fk}_i,\mathsf{tag}^*}(r_i^*)\}_{i\in[d]}\big)$. Output $r^*$.

First, for any fixed $i \in [d]$, the probability over $r_i^* \leftarrow \{0,1\}^n$ that $F_{\mathsf{fk}_i,\mathsf{tag}^*}(r_i^*) = F_{\mathsf{fk}_i,\mathsf{tag}^*}(r_i)$ is at least $\frac{2^\ell}{2^n}$. This is because in the construction of Theorem 2, each output $F_{\mathsf{fk}_i,\mathsf{tag}^*}(r_i)$ has at least $2^\ell$ preimages for all $r_i \in \{0,1\}^n$ and all $\mathsf{fk}_i \leftarrow \mathsf{LossyGen}(\mathsf{tag}^*, r_i)$.

As a result, the probability that this happens for all $i \in [d]$ is at least $\frac{2^{\ell\cdot d}}{2^{n\cdot d}} = \frac{2^{\ell\cdot d}}{2^\rho}$. In other words, $\mathcal{I}$ correctly guesses all values $\{F_{\mathsf{fk}_i,\mathsf{tag}^*}(r_i^*)\}_{i\in[d]}$ with probability $\frac{2^{\ell\cdot d}}{2^\rho}$, and therefore, by correctness of $\mathcal{P}$, $\mathcal{I}$ correctly outputs $r$ with probability at least $\frac{2^{\ell\cdot d}}{2^\rho} \cdot \epsilon$, contradicting the security of $\mathsf{TDF}$.

*Remark 3.* One can interpret Lemma 1 as showing that the encryption scheme with randomness recovery $\mathsf{CPA}$ of [HKW20] is resilient to some particular form of leakage, namely, leakage generated by the T-A**I**BO $F$ of Section 4.2 (in lossy mode). Most notably, we point out that (1) the leakage is over the encryption

randomness, which we can usually fix generically using strong seeded extractors (but does not work here as it would not preserve randomness recovery) and (2) it is a particular form of *entropy-bounded* leakage ( [NS09, DHLW10]) in the sense that the size of the leakage is quite bigger than the entropy loss, namely, $|F_{\mathsf{fk},\mathsf{tag}}(r)| > \ell$. This is as opposed to the perhaps more common form of *bounded-size* leakage.

*Remark 4 (Usage of a Specific T-A**I**BO).* Lemma 1 is stated as using the specific leakage function, defined by the specific T-A**I**BO of Section 4.2. In fact, the only property we use is that, for all fixed input, the leakage function can be guessed with slightly better than trivial. As a result, any T-A**I**BO satisfying this property can be used in Lemma 1, and consequently in our construction Appendix B.2.

*Remark 5 (Running Time VS Success Probability).* Lemma 1 and its proof involve algorithms that have low running time but small success probability, namely, we build a time $T \cdot \mathrm{poly}(\lambda)$ inverter with success probability $\frac{2^{\ell \cdot d}}{2^{\rho}} \cdot \epsilon$. We note here that we can also build a time $T \cdot \frac{2^{\rho}}{2^{\ell \cdot d}} \cdot \mathrm{poly}(\lambda)$ inverter that succeeds with probability $\Omega(\epsilon)$. This is by having the inverter attempt $\frac{2^{\rho}}{2^{\ell \cdot d}}$ samples for the values $r_i^*$. Then, with constant probability (over the randomness of these samples alone), one of them is right, conditioned on which the predictor $\mathcal{P}$ succeeds in inverting with probability $\epsilon$.

This, in turn allows to alternatively instantiate Theorem 22 with a $\mathsf{TDF}$ such that no time $T = 2^{\rho-(\ell d-n)}$ adversary succeeds in inverting $\mathsf{TDF}$ with non-negligible probability.

### B.2 CCA-secure Encryption from T-AIBOs and Trapdoor Functions

We now describe our construction of a CCA-secure encryption scheme. Let $d = d(\lambda)$, $n = n(\lambda)$, $t = t(\lambda)$, $\rho = d \cdot n$. We will use the following components:

- the T-A**I**BO $(\mathsf{InjectiveGen}, \mathsf{LossyGen}, F)$ of Section 4.2 with input length $n$ and tag space $\{0,1\}^t$, satisfying $\ell$-lossiness from Theorem 2;
- the encryption scheme $(\mathsf{CPA.KeyGen}, \mathsf{CPA.Enc}, \mathsf{CPA.Dec}, \mathsf{CPA.Recover})$ of Appendix B.1, with encryption randomness space $\{0,1\}^{\rho}$ and ciphertext size $\{0,1\}^{\tau}$, which can be built from any trapdoor function $\mathsf{TDF}$ with input length $\rho$. In particular it satisfies *randomness recovery* (Definition 13), and satisfies some special form of leakage resilience with respect to the T-A**I**BO above (Lemma 1). More precisely, we will assume that no time $T = 2^n \cdot \mathrm{poly}(\lambda)$ adversary can invert the underlying $\mathsf{TDF}$ with probability $\frac{2^d}{2^{\rho}} \cdot \epsilon$ for any non-negligible $\epsilon$.
- a one-time strongly unforgeable signature $(\mathsf{Sig.KeyGen}, \mathsf{Sig.Sign}, \mathsf{Sig.Verify})$ with message space $\{0,1\}^{\tau+d \cdot m}$, where $\tau$ is the size of ciphertexts from $\mathsf{CPA}$, and with verification keys of size (at most) $t$.

We now describe our CCA-secure encryption scheme:

- $\mathsf{KeyGen}(1^\lambda)$: Set $(\mathsf{CPA.pk}, \mathsf{CPA.sk}) \leftarrow \mathsf{CPA.KeyGen}$ and compute, for all $i \in [d]$: $\mathsf{fk}_i \leftarrow \mathsf{InjectiveGen}$. Set $\mathsf{pk} = (\mathsf{CPA.pk}, \{\mathsf{fk}\}_{i\in[d]})$ and $\mathsf{sk} = \mathsf{CPA.sk}$ and output $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{Enc}(\mathsf{pk}, \mathsf{msg})$: Sample $r \leftarrow \{0,1\}^\rho$. Compute

$$C = \mathsf{CPA.Enc}(\mathsf{CPA.pk}, \mathsf{msg}\,;\, r).$$

  Sample $(\mathsf{Sig.vk}, \mathsf{Sig.sk}) \leftarrow \mathsf{Sig.KeyGen}$. Parse $r$ as $r = (r_1 \| \cdots \| r_d)$ where $r_i \in \{0,1\}^n$. Parsing the verification key of the signature as a tag for the T-A**I**BO, compute, for $i \in [d]$:

$$y_i = F_{\mathsf{fk}_i, \mathsf{Sig.vk}}(r_i).$$

  Compute

$$\sigma = \mathsf{Sig.Sign}(\mathsf{Sig.sk}, (C \| y_1 \| \cdots \| y_d)),$$

  and output

$$\mathsf{ct} = (\mathsf{Sig.vk}, C, \{y_i\}_{i\le d}, \sigma).$$

- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$: Parse $\mathsf{ct}$ as $(\mathsf{Sig.vk}, C, \{y_i\}_{i\le d}, \sigma)$. Output $\perp$ if $\mathsf{Sig.Verify}(\mathsf{Sig.vk}, (C \| y_1 \| \cdots \| y_t), \sigma) = 0$.
  Else compute $(\mathsf{msg}, r) \leftarrow \mathsf{CPA.Dec}(\mathsf{ct})$, using the randomness recovery property of the CPA-encryption scheme. Parse $r$ as $(r_1 \| \cdots \| r_d)$ where $r_i \in \{0,1\}^n$ for all $i \in [d]$.
  If $\mathsf{CPA.Enc}(\mathsf{msg}, r) = C$, and if $F_{\mathsf{fk}_i, \mathsf{Sig.vk}}(r_i) = y_i$, output $\mathsf{msg}$. Otherwise output $\perp$.

**Claim 10.** *Suppose that no time $T = 2^n \cdot \mathrm{poly}(\lambda)$ adversary can invert the* TDF *underlying* CPA *with probability $\frac{2^d}{2^\rho} \cdot \epsilon$ for any non-negligible $\epsilon$, that T-A**I**BO is the one constructed in Section 4.2, and* $(\mathsf{Sig.KeyGen}, \mathsf{Sig.Sign}, \mathsf{Sig.Verify})$ *is one-time strongly unforgeable. Then* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is CCA-secure.*

As noted in Remark 4, one could replace the specific T-A**I**BO of Section 4.2 with one satisfying a generic non-trivial guessing property in lossy mode.

*Proof.* We consider a sequence of hybrids.

*Hybrid $H_0$.* This is the standard CCA experiment.

*Hybrid $H_1$.* The challenger changes how it answers decryption queries and the challenge ciphertext. It now picks $(\mathsf{Sig.vk}^*, \mathsf{Sig.sk}^*) \leftarrow \mathsf{Sig.KeyGen}$ at the beginning of the experiment, and answers $\perp$ to any decryption query whose first component is $\mathsf{Sig.vk}^*$.

The challenge ciphertext is generated using $(\mathsf{Sig.vk}^*, \mathsf{Sig.sk}^*)$, namely, the other components of the challenge ciphertext are computed as:

$$C^* = \mathsf{CPA.Enc}(\mathsf{CPA.pk}, \mathsf{msg}_b^*\,;\, r^*),$$
$$y_i^* = F_{\mathsf{fk}_i, \mathsf{Sig.vk}^*}(r_i^*),$$
$$\sigma^* = \mathsf{Sig.Sign}(\mathsf{Sig.sk}^*, (C \| y_1^* \| \cdots \| y_d^*),$$

where $r^* = (r_1^* \| \cdots \| r_d^*) \leftarrow \{0,1\}^\rho$, and $\mathsf{msg}_0^*, \mathsf{msg}_1^*$ are the challenge messages sent by the adversary. The challenge ciphertext is $\mathsf{ct}^* = (\mathsf{Sig.vk}^*, C^*, \{y_i^*\}_{i\in[d]}, \sigma^*)$.

*Hybrid $H_2$.* The challenger changes how it generates the public key of the scheme, and more precisely how it samples the T-AIBO keys $\mathsf{fk}_i$. It now samples $r^* = (r_1^* \| \dots \| r_d^*) \leftarrow \{0,1\}^\rho$ and computes $\mathsf{fk}_i$ as $\mathsf{fk}_i \leftarrow \mathsf{LossyGen}(\mathsf{Sig.vk}^*, r_i^*)$.

*Hybrid $H_3$.* The challenger changes how it answers decryption queries. Given a ciphertext $\mathsf{ct} = (\mathsf{Sig.vk}, C, \{y_i\}_{i \le d}, \sigma)$, the challenger:

1. Checks that $\mathsf{Sig.Verify}(\mathsf{Sig.vk}, (C \| y_1 \| \cdots \| y_d), \sigma) = 1$, and outputs $\perp$ otherwise;
2. For all $i \in [d]$, it enumerates over all $r_i \in \{0,1\}^n$ the values $F_{\mathsf{fk}_i, \mathsf{Sig.vk}}(r_i)$, and outputs the first such $r_i$ such that $F_{\mathsf{fk}_i, \mathsf{Sig.vk}}(r_i) = y_i$. Otherwise it outputs $\perp$;
3. Using the randomness recovery property of $(\mathsf{CPA.KeyGen}, \mathsf{CPA.Enc}, \mathsf{CPA.Dec})$, it recovers $m \leftarrow \mathsf{Recover}(\mathsf{pk}, C, r)$ where $r = (r_1 \| \cdots \| r_d)$, and checks that $\mathsf{ct} = \mathsf{CPA.Enc}(\mathsf{CPA.pk}, \mathsf{msg}; r)$, and outputs $\perp$ otherwise;
4. Outputs $\mathsf{msg}$.

*Hybrid $H_4$.* The challenger changes how it produces the challenge ciphertext. It now computes $C = \mathsf{CPA.Enc}(\mathsf{CPA.pk}, \mathbf{0}; r^*)$.

We now prove that successive hybrids are indistinguishable.

**Claim 11.** *Suppose* Sig *is one-time strongly unforgeable. Then the views of the adversary in $H_0$ and $H_1$ are indistinguishable.*

*Proof.* The views in $H_0$ and $H_1$ differ exactly when the adversary makes a decryption query with respect to the verification key of the challenge ciphertext $\mathsf{Sig.vk}^*$ such that the associated signature $\sigma$ verifies. We distinguish several cases.

- The ciphertext of such a query uses the same signature $\sigma^*$ and message $(C^* \| y_1^* \| \cdots \| y_t^*)$ as the challenge ciphertext, namely queries the challenge ciphertext. By definition of the CCA experiment, this corresponds to the case where the adversary guesses the challenge ciphertext and queries it before receiving the actual challenge ciphertext from the challenger. This only happens with negligible probability.
- Otherwise, the ciphertext of such a query uses a different signature $\sigma \neq \sigma^*$. We argue that such an adversary then induces an adversary against the one-time strongly unforgeability of Sig. A reduction samples the parameters $(\mathsf{pk}, \mathsf{sk})$ of the real scheme, and uses them to answer decryption queries. It then receives $\mathsf{msg}_0, \mathsf{msg}_1$ from the adversary, and chooses a random bit $b \leftarrow \{0,1\}$. It interacts with the unforgeability experiment challenger to receive a verification key $\mathsf{vk}^*$. As in the real scheme, computes $C^* = \mathsf{CPA.Enc}(\mathsf{CPA.pk}, m_b; r^*$, and $y_i^* = F_{\mathsf{fk}_i, \mathsf{Sig.vk}}(r_i^*)$ where $r^* = (r_1^* \| \cdots \| r_d^*) \leftarrow \{0,1\}^\rho$. It queries the challenger for the unforgeability experiment with message $m = (C^* \| y_1^* \| \cdots \| y_d^*)$, and receives a signature $\sigma^*$. It sets the challenge ciphertext as $\mathsf{ct}^* = (\mathsf{Sig.vk}^*, C^*, \{y_i\}_{i \in [d]}, \sigma^*)$.
  Now whenever the adversary for the CCA game decryption query $\mathsf{ct} \neq \mathsf{ct}^*$ containing $\mathsf{Sig.vk}^*$ such that the associated signature verifies, either the message or the signature differs from the challenge ciphertext (by analysis of the

previous case above). In either case, such a decryption query then induces a forgery for $\mathsf{Sig}$.

**Claim 12.** *Suppose* $(\mathsf{InjectiveGen}, \mathsf{LossyGen}, F)$ *satisfies indistinguishability. Then the views of the adversary in* $H_1$ *and* $H_2$ *are indistinguishable.*

*Proof.* This follows by switching how the keys $\mathsf{fk}_i$ are generated, one by one, from $\mathsf{InjectiveGen}$ to $\mathsf{LossyGen}(\mathsf{vk}^*, r_i^*)$, over $i \in [d]$. The reduction between two consecutive sub-hybrids picks $\mathsf{tag}^* = \mathsf{Sig.vk}^*$, receives $s^* = r_i^* \in \{0,1\}^n$ and $\mathsf{fk}$, and uses $\mathsf{fk}$ in the public key and $r_i^*$ as the appropriate randomness bloc to generate the challenge ciphertext.

**Claim 13.** *Suppose* $(\mathsf{InjectiveGen}, \mathsf{LossyGen}, F)$ *satisfies injectivity on injective branches, and that* $\mathsf{CPA}$ *is statistically correct. Then the views of the adversary in* $H_2$ *and* $H_3$ *are distributed within negligible statistical distance.*

*Proof.* By the injectivity property of the T-A**I**BO, one can check that in both hybrids, a decryption queries on $\mathsf{ct} = (\mathsf{Sig.vk}, C, \{y_i\}_{i\in[d]}, \sigma)$ does not output $\perp$ if and only if:

- $\mathsf{Sig.Verify}(\mathsf{vk}, (C\|y_1\|\cdots\|y_d), \sigma) = 1$;
- $r_i$ is the only input such that $F_{\mathsf{fk}_i,\mathsf{Sig.vk}}(r_i) = y_i$ (where we used $\mathsf{Sig.vk} \neq \mathsf{Sig.vk}^*$ to ensure injectivity of $F_{\mathsf{fk}_i,\mathsf{Sig.vk}}(\cdot)$);
- $\mathsf{msg}$ is correctly recovered as $\mathsf{msg} \leftarrow \mathsf{Recover}(\mathsf{pk}, C, m)$;
- $C = \mathsf{CPA.Enc}(\mathsf{CPA.pk}, \mathsf{msg}; r)$.

In particular, the hybrids differ whenever $F_{\mathsf{fk}_i,\mathsf{Sig.vk}}$ is not injective or $\mathsf{Recover}$ is not correct, which both happen with negligible probability over the randomness of $\mathsf{fk} \leftarrow \mathsf{InjectiveGen}(1^\lambda)$ and $(\mathsf{CPA.pk}, \mathsf{CPA.sk}) \leftarrow \mathsf{CPA.KeyGen}(1^\lambda)$, respectively.

**Claim 14.** *Suppose* $\mathsf{CPA} = (\mathsf{CPA.KeyGen}, \mathsf{CPA.Enc}, \mathsf{CPA.Dec})$ *is instantiated with a trapdoor function* $\mathsf{TDF}$ *such that no time* $T = 2^n \cdot \mathrm{poly}(\lambda)$ *adversary can invert* $\mathsf{TDF}$ *with probability* $\frac{2^d}{2^\rho} \cdot \epsilon$ *for any non-negligible* $\epsilon$*. Then the views of the adversary in* $H_3$ *and* $H_4$ *are indistinguishable.*

*Proof.* We show that any polynomial-time distinguisher between $H_3$ and $H_4$ induces a time $d \cdot 2^n \cdot \mathrm{poly}(\lambda)$ against the distribution of Lemma 1. The intuition is that Lemma 1 ensures that $\mathsf{CPA}$ is leakage resilient when the leakage is computed using the T-A**I**BO $F$ (in lossy mode).

Our reduction samples $(\mathsf{Sig.vk}^*, \mathsf{Sig.sk}^*)$ and sets $\mathsf{tag}^* = \mathsf{Sig.vk}^*$. It receives a sample from the distribution of Lemma 1

$$\big(\{\mathsf{fk}_i\}_{i\in[d]}, \mathsf{CPA.pk}, \mathsf{TDF.Eval}_{\mathsf{pk}}(r^*), \{F_{\mathsf{fk}_i,\mathsf{tag}^*}(r_i^*)\}_{i\in[d]}, b\big),$$

where $r = (r_1\|\ldots,\|r_d) \leftarrow \{0,1\}^\rho$, $\mathsf{fk}_i \leftarrow \mathsf{LossyGen}(\mathsf{tag}^*, r_i)$, $\mathsf{pk} \leftarrow \mathsf{TDF.Setup}$, and $b \in \{0,1\}$.

The reduction sets the public key as $\mathsf{pk} = (\mathsf{CPA.pk}, \{\mathsf{fk}\}_{i\in[d]})$, and sets the challenge ciphertext as

$$C^* = \mathsf{CPA.Enc}(\mathsf{CPA.pk}, \mathsf{msg}^* ; r^*) = (\mathsf{TDF.Eval}_{\mathsf{pk}}(r^*), b),$$
$$y_i^* = F_{\mathsf{fk}_i, \mathsf{Sig.vk}^*}(r_i^*),$$
$$\sigma^* = \mathsf{Sig.Sign}(\mathsf{Sig.sk}^*, (C \| y_1^* \| \cdots \| y_d^*),$$

It answers decryption queries as in Hybrid $H_3$, so that each decryption query is answered in time $d \cdot 2^n$. This simulates perfectly Hybrid $H_3$ if $b = \mathsf{hc}(r) \oplus m$, and Hybrid $H_4$ if $b = \mathsf{hc}(r)$, and in particular any distinguisher between $H_3$ and $H_4$ induce a time $d \cdot 2^n$ distinguisher for the distribution of Lemma 1. Then, Lemma 1 implies that such a distinguisher implies a time $2^n \cdot \mathrm{poly}(\lambda)$ inverter against $\mathsf{TDF}$ with success probability $\frac{2^d}{2^\rho} \cdot \epsilon$ for some non-negligible $\epsilon$.

This finishes the proof of CCA-security of $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$.

Theorem 22 follows from Claim 10, using the fact that one-time strongly unforgeable signatures are implied by one-way functions (which are implied by either PRGs or trapdoor functions).