Deterministic Public-Key Encryption under Continual Leakage

Venkata Koppula[*] Omkant Pandey[†] Yannis Rouselakis[‡] Brent Waters[§]

Abstract

Deterministic public-key encryption, introduced by Bellare, Boldyreva, and O'Neill (CRYPTO 2007), is an important technique for searchable encryption; it allows quick, logarithmic-time, search over encrypted data items. The technique is most effective in scenarios where frequent search queries are performed over a huge database of unpredictable data items. We initiate the study of deterministic public-key encryption (D-PKE) in the presence of *leakage*. We formulate appropriate security notions for leakage-resilient D-PKE, and present constructions that achieve them in the standard model. We work in the *continual* leakage model, where the secret-key is updated at regular intervals and an attacker can learn arbitrary but bounded leakage on the secret key during each time interval. We, however, do not consider leakage during the updates. Our main construction is based on the (standard) linear assumption in bilinear groups, tolerating up to $0.5 - o(1)$ fraction of arbitrary leakage. The leakage rate can be improved to $1 - o(1)$ by relying on the SXDH assumption.

At a technical level, we propose and construct a "continual leakage resilient" version of the *all-but-one* lossy trapdoor functions, introduced by Peikert and Waters (STOC 2008). Our formulation and construction of leakage-resilient lossy-TDFs is of independent general interest for leakage-resilient cryptography.

1 Introduction

The notion of *semantic security* for public key encryption schemes was introduced in the seminal work of Goldwasser and Micali [GM84]. While this strong notion of security is desirable in many applications, it requires that the encryption algorithm must be a random process. This creates a significant performance bottleneck if, for example, one wants to perform fast search over many encrypted data items. To address this issue, Bellare, Boldyreva, and O'Neill [BBO07] initiated the study of deterministic public-key encryption (D-PKE) schemes. In D-PKE schemes, the encryption algorithm is required to be a deterministic function of the message. Consequently, D-PKE cannot satisfy any meaningful notion of security for low-entropy plaintext distributions. Bellare et al. demonstrated that a strong notion of security can in fact be realized for relatively high-entropy plaintext distributions. Several follow up works then further investigated security notions for deterministic encryption and presented standard model constructions [BFOR08, BFO08, O'N10, BS11, MPRS12, FOR12, RSV13].

Deterministic encryption is a promising technique for building "searchable encryption" [PRZB11, PRZB12]. It is most effective in scenarios where frequent search queries are performed over a huge database of *unpredictable*, data items (e.g., credit card numbers). This is in fact the ideal setting for deterministic encryption: on one hand, the "hard-to-guess" nature of credit-card numbers ensures that they are well protected even if encryption is deterministic; on the other hand, logarithmic

[*]The University of Texas at Austin. Email: `kvenkata@cs.utexas.edu`

[†]University of California, Berkeley. Email: `omkant@berkeley.edu`.

[‡]Microsoft, Redmond. Email: `johnysrouss@gmail.com`.

[§]The University of Texas at Austin. Email: `bwaters@cs.utexas.edu`

search-time ensures good response-time even if the database is potentially huge and search queries are rather frequent.

We initiate a study of deterministic PKE in the presence of leakage attack where an adversary can learn partial but important information about the secret key of the system (e.g., by means of insider attacks, or side channel attacks [Koc96, AK97, QS01, OST06]). Existing deterministic PKE schemes are not resilient to leakage and assume that the adversary only has black-box access to the decryption box.

We present a thorough study of leakage-resilient D-PKE. We adapt existing security notions for deterministic PKE to the *continual leakage* model [DHLAW10, BKKV10] and present constructions that achieve them.

Let us note that although several leakage-resilient schemes for *randomized* PKE are known [DP08, AGV09, DKL09, ADN$^+$10, NS09, BKKV10, DHLAW10, HLWW13], they have no direct implication to determinisitc PKE. This is because the security of randomized PKE crucially relies on the *randomness* of encryption even in the leakage setting and such randomness is simply not present in deterministic PKE. In general—*even without leakage*—there is no direct way of obtaining deterministic PKE schemes from the randomized ones in the standard model; the techniques for constructing deterministic PKE are usually quite different.

The *continual memory leakage* (CML) model was introduced in [DHLAW10, BKKV10]. In the context of public-key encryption, we envision a system with a *fixed* public-key pk, along with a (variable) secret key sk which is "refreshed" or "updated" at regular time intervals. In each time interval, the adversary can issue a leakage query of its choice, in the form of a polynomial-time computable function $\mathcal{L}$, and learn $\mathcal{L}(sk)$. The adversary can repeat this process for polynomially many time intervals, issuing queries of the form $\mathcal{L}_i$, and learning $\mathcal{L}(sk_i)$ in the i-th interval. To prevent trivially leaking the whole key, the model asserts that the size of all leakage answers in the i-th intervals is bounded by $\rho.|sk_i|$ for every i, where $\rho \in [0, 1)$ is the leakage parameter of the system and sk_i is the secret-key in the i-th interval. It is also required that every sk_i should correctly decrypt the ciphertexts under pk, and be roughly of the same size as the initial key of the system. The higher the ρ for a scheme, the higher the amount of leakage it can tolerate.[1]

The CML model is indeed a very powerful model since it allows the attacker to potentially learn unbounded leakage on the system's secret memory. In addition to constructions of randomized PKE mentioned above, leakage-resilient schemes for several tasks in a variety of leakage models are now known, e.g., digital signatures [LW10, BSW11, LLW11], identity-based encryption [BKKV10, CDRW10, LRW11], interactive proofs [GJS11, Pan14, AGP14], secure computation [FRR$^+$10, BGJK12, GR12], and so on. We remark that the study of leakage-resilient cryptography was initiated in [DP08, MR04, ISW03] as an attempt to provide an *algorithmic* defense against side-channel attacks [Koc96, AK97, QS01, OST06]. Renauld, Standaert, Veyrat-Charvillon, Kamel, and Flandre [RSV$^+$11] highlight several difficulties in formalizing an appropriate model of leakage for real-world side-channel attacks, and argue that often an algorithmic defense is not possible since the key might have been completely compromised. In such settings, we cannot do anything except for developing alternative methods such as those at the hardware level. However, when the adversary *is* limited to side-channel attacks that do not fully compromise the system, the continual leakage model is essentially as good a model as possible.

[1]We note that in our model no leakage is allowed during *update* phase. However, the most general model allows leakage during the update phase as well.

Our results. Our goal is to obtain standard model constructions of deterministic PKE which can deliver meaningful security under the CML attack. The security notions for deterministic PKE have evolved over time. Bellare et al. [BBO07] proposed the notions of PRIV1-IND and PRIV-IND security; the former is the most basic notion of security, while the latter was shown in [BFOR08, BFO08] to be one of the strongest notions. In the special setting when plaintext distributions have sufficient *block-wise min-entropy*, these two notions are actually equivalent. Bellare et al. [BBO07] argued that in the general setting, the plaintext distributions cannot depend on the public-key of the system. However, under special constraints over plaintext distributions, this restriction may not be necessary [BS11, RSV13].

In this work, we stick to the original setting of [BBO07], and reformulate PRIV1-IND and PRIV-IND security in the presence of CML attacks. We then construct a deterministic PKE scheme satisfying the PRIV1-IND security under the CML attack. Our scheme is based on the (standard) *linear* (a.k.a. "matrix DDH") assumption [BBS04, NS09] in bilinear groups, and it can tolerate a leakage-rate up to $\rho = 0.5 - o(1)$. A simpler variant of this scheme has better system parameters, and can tolerate an almost optimal leakage rate of $\rho = 1 - o(1)$; however, it is based on the (stronger) SXDH assumption.

To construct our deterministic PKE, we formulate and construct a "continual leakage resilient" version of lossy trapdoor functions, abbreviated as CLR-LTDF. Lossy trapdoor functions were introduced by Peikert and Waters [PW08], and have found a vast number of applications in cryptography. We actually work with the more general notion of lossy TDFs, namely all-but-one (ABO) functions, since the simpler definition (consisting of only two families) cannot tolerate even 1 bit of leakage. We believe that our formulation of CLR-LTDF is of independent general interest especially with regard to constructing other leakage-resilient schemes.

We remark that unlike the standard setting where lossy TDFs almost immediately imply D-PKE [BFO08], CLR-LTDFs do not immediately imply a leakage-resilient D-PKE. The leakage setting is more challenging and the proof that such a reduction is possible, is not straight forward.

1.1 An overview of our approach

Bellare et al. [BBO07] show that, in the random-oracle model [BR93], a semantically secure (randomized) PKE also implies a PRIV-IND secure deterministic PKE; the reduction simply replaces the randomness of encryption by $H(m)$ where H is a random-oracle and m is the message to be encrypted. By using a LR (randomized) PKE in this reduction, we immediately get a LR D-PKE. However, in the standard model, no such general reduction is known. In general, due to the deterministic nature of encryption, D-PKE generally require their own set of techniques.

A prominent technique for constructing deterministic PKE (in the standard model) is based on *lossy trapdoor functions* [PW08]; it was given by Boldyreva, Fehr, and O'Neill [BFO08]. Recall that, lossy TDFs define two function families $\{F_{\text{inj}}\}$ and $\{F_{\text{lossy}}\}$. Functions in the first family are always *injective* and can be efficiently inverted using a trapdoor. Functions in the other family are always *lossy*: meaning that the range-size of every function in $\{F_{\text{lossy}}\}$ is much smaller than its domain-size. Therefore, functions in the second family necessarily loose a lot of information about their input. In addition, these two families are computationally indistinguishable: it is hard to decide whether a (properly sampled) function belongs to the injective family or the lossy family. Boldyreva et al. [BFO08] observe that if the lossy mode also acts as a *universal* hash function then the functions from the injective family (of a lossy TDF) act as a PRIV1-IND secure deterministic-PKE. Furthermore, following [DS05], they show that even if the lossy mode is not universal, it still leads to a secure scheme *provided* that the message is first permuted using a pairwise independent permutation. They prove this by extending the crooked LHL of [DS05] to work with lossy functions and average conditional min-entropy (see lemma 2).

A natural idea is to suitably adapt this approach to the leakage setting. Unfortunately, lossy TDFs cannot be leakage-resilient as defined: just one bit of leakage on the trapdoor suffices to tell injective functions from the lossy ones. To make this approach work, we first need to re-formulate the notion of lossy TDFs in the leakage setting and then suitably modify the approach of [BFO08] to obtain a deterministic PKE.

Leakage-resilient lossy TDFs. Since lossy TDFs cannot be leakage-resilient as defined, we work with *all-but-one* (ABO) functions also introduced in [PW08]. ABO functions define only one family $\{F\}$ where each $f \in F$ takes two inputs. The first input is called a *branch* b taken from a branch space $\mathcal{B}$. As the name suggests, there exists a unique branch $b^* \in \mathcal{B}$ such that the single input function $F(b, \cdot)$ is lossy when $b = b^*$ and injective otherwise. We consider a notion similar to ABO functions. Specifically, we consider a family of functions $\{F\}$ which take two inputs where the first input is a branch b. We require that at least one branch b defines a lossy function $F(b, \cdot)$, and the fraction of all lossy branches is negligible. All other branches define an injective function.

Intuitively, leakage resilience for our functions, should mean that given $(f, \mathcal{L}_f(t), b)$, where is t is the trapdoor, it is hard to decide whether b is lossy or injective; here $\mathcal{L}_f$ is the leakage function which can depend on f (but not b, since otherwise we will have the same problem as before). We note that it is of independent interest to consider such functions under various models for leakage $\mathcal{L}_f$. However, motivated by our application of deterministic PKE, we will consider the most demanding CML model. Since the CML model requires that the trapdoor should refreshed or updated after each time interval, our functions will have an *update* algorithm in addition to usual ones for ABO.

The attack model for our functions will then work as follows. Once the description of f is fixed, the adversary will be able to ask (bounded) leakage during each time-interval; the trapdoor for the function f will be *updated* after each time interval. Once the leakage is complete, the adversary will enter a challenge phase in which it will be given either a lossy branch b^* or an injective branch $b \neq b^*$; the adversary wins if it successfully guesses the type of the branch. The adversary is not allowed any queries once the challenge branch is given. The formal description appears in later sections.

As mentioned earlier, we call such functions *continual leakage-resilient lossy-trapdoor functions* or CLR-LTDF. We construct CLR-LTDF under the linear assumption tolerating a leakage fraction of $\rho = 0.5 - o(1)$. A simpler variant of this construction based on SXDH assumption can tolerate almost optimal leakage of $\rho = 1 - o(1)$.

Achieving leakage-resilient D-PKE from CLR-LTDF. Although our formulation of CLR-LTDF seems natural, it remains to be seen if it can prove useful in constructing D-PKE. Let us see if we can use an approach along the lines of Boldyreva et al. [BFO08].

Suppose that we are given a family of CLR-TDFs. The functions in the family require a branch for evaluation. We need to find a *deterministic* method to sample the branch. If a branch is chosen and provided with public-parameters, we will not have any leakage-resilience. The adversary can simply check if the branch is lossy or injective via leakage queries. A better idea, following [BFO08], would be to let the branch $b = h(m)$ and then encrypt $\pi(m)$ where m is the message to be encrypted, h is a universal hash function, and π a pairwise independent permutation; both (h, π) are sampled at the time of setup. If m has sufficient entropy the branch looks random due to (standard) LHL; further if m has enough entropy *conditioned* on $(h, h(m))$, we might hope to use the analysis from [BFO08] (which relies on "generalized crooked LHL") to argue security.[2]

[2]Note that here it is important that distribution of m does not depend on h, π. This is indeed the case since (h, π)

Unfortunately, the analysis from [BFO08] does not quite work since it crucially relies on the fact that the family is a lossy TDF *without branches.* (If ABO functions are used, then a single branch must be chosen as part of the public-parameters, and used for all evaluations for their analysis to work.) In contrast, in our proposed scheme, the branch changes for almost every sampled message. This results in two main difficulties. Consider the reduction in which the proof will try to reduce the security of our proposed construction to that of the CLR-LTDF. That is, the reduction attempts to correctly guess whether a challenge branch b' is injective or lossy with the help of an adversary A who breaks our proposed scheme. The reduction will somehow need to use b' to create a correctly generated challenge ciphertext c. For example, in the simplest type of reduction, we may try to ensure that c is an encryption under the branch b' for some message, say m. Then, the reduction must ensure that: (1) $h(m) = b'$, and (2) m comes from one distribution if b' is injective and from the other distribution if it is lossy.

It turns out rather non-trivial to show that we can design such a reduction and the proposed construction indeed works. However, making this construction work requires us to consider a slightly strengthened version of CLR-LTDFs where the challenge branches can be sampled using an arbitrary, possibly adversarial, algorithm as long as the sampling results in independent and correctly distributed branches. The security is then required to hold even in the presence of some auxiliary information about the challenge branch.

1.2 How to construct CLR-LTDF

We now present an overview of our construction of CLR-LTDFs. Our starting point is the ABO construction of [PW08] which works as follows. It samples a $n \times n$ matrix $\mathbf{A}$ whose entries are, roughly speaking, (ElGamal) encryptions of 0. The function description is then set to $g^{\mathbf{M}}$ where $\mathbf{M} = \mathbf{A} + b^*\mathbf{I}$, $\mathbf{I}$ denotes the identity matrix, and b^* is lossy branch. The function evaluation on $(b, \mathbf{x})$ is $g^{(\mathbf{M}-b\mathbf{I})\mathbf{x}}$, which is easily inverted if $\mathbf{x} \in \{0,1\}^n$, given $\mathbf{A}, b, b^*$. To ensure lossiness when $b = b^*$, the randomness of the ElGamal encryptions are "correlated" in a special manner.

A natural idea is to replace the ElGamal encryption with an appropriate continual LR PKE scheme. The hope is that leakage-resilience of ABO can now be reduced to leakage-resilience of the PKE in use. The central difficulty such a reduction faces is as follows. The reduction needs to provide the adversary, say A, the description of a function f from the family in the beginning, and a challenge branch b' in the end. Clearly, whether b' is injective or lossy, should somehow depend on whether the challenge ciphertext c obtained from the PKE-challenger encrypts 0 or 1.[3] Since c is not known ahead of time, the reduction must be able to find a b' when c becomes known, yet this b' should make the function injective or lossy depending on what c encrypts.

We do not know if such a black-box reduction is possible in general. However, it might be possible to make this approach work by relying on specific constructions of LR PKE. Our construction uses a similar approach and uses BKKV encryption instead of ElGamal. A key property of this encryption scheme is that it supports (almost) additive homomorphic encryptions. This allows us to use the branches as a one-time pad to mask the diagonal entries. However, we still cannot obtain an injective or lossy branch from a challenge ciphertext of BKKV in a black-box manner. Instead, we directly work with "matrix DDH" challenger, and use the ideas from BKKV, to directly prove that our construction is leakage-resilient. More specifically, our reduction directly works with the matrix DDH challenger, but relies on the structure of BKKV encryption and the fact that *random subspaces are leakage resilient* (Theorem 2.1, [BKKV10]) to answer the leakage

are part of the public-key and m is not allowed to depend on public-key in our setting.

[3]For the PW construction, we will need to use n such ciphertexts, one for each diagonal entry; this can be handled using a hybrid argument.

queries of the adversary. The reduction only knows a key from a random subspace of the entire key space, and therefore can fail to decrypt some special ciphertexts. These special cipertexts will correspond to the DDH challenges, and used to define the challenge branch b'. A description of BKKV encryption scheme is given in the preliminaries.

Related work on lossy TDFs. There has been a significant follow up work on extending lossy TDFs such as all-but-N functions defined by Hemnway et al. [HLOV11], all-but-many functions defined by Hofheinz [Hof12], and identity-based lossy TDFs defined by Bellare et al. [BKPW12]. Likewise, in addition to the constructions in these works, several other works have presented constructions of (standard) lossy TDFs based on a variety of assumptions [FGK$^+$10, Wee12]. Qin et al. [QLCC13] propose an alternate type of leakage-resilient lossy-TDF in which a master-key mk is selected first; injective/lossy functions f are chosen later, and mk generates trapdoors for injective f. Leakage is allowed only on mk, and no leakage is allowed after f is fixed. It is not clear how to use this version for our application of deterministic-PKE.

2 Preliminaries

Standard notation. We use uppercase roman letters $A, X, \ldots$ for representing sets and matrices, lowercase roman letters to denote elements of a set $x \in X$, and bold for vectors $\mathbf{x} \in X^n$. When X is a set, $x \overset{\$}{\leftarrow} X$ denotes a uniformly sampled $x \in X$. Notation $\langle \mathbf{x}, \mathbf{y} \rangle$ represents standard scalar product of vectors $\mathbf{x}$ and $\mathbf{y}$. If A is a matrix then $A\mathbf{x}$ denotes the standard matrix product where $\mathbf{x}$ is seen as a column vector (with scalar coordinates). The transpose of a matrix A is denoted by A^{T}. For vectors $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$, we denote by $\mathsf{span}(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ and $\mathsf{kernel}(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ the linear-span and the kernel space of these vectors.

Sets $\mathbb{N}, \mathbb{R}, \mathbb{Z}$ and $\mathbb{Z}_p$ denote the sets of natural numbers, real numbers, integers, and integers modulo p respectively. We let $\mathsf{Rank}_r^{m \times n}(S)$ be the set of all $m \times n$ matrices of rank r whose entries are from S, whenever such a rank is well defined, e.g., when $S = \mathbb{Z}_p$. Given any function f, $|f(\cdot)|$ represents the size of image space of f. A function $\mu : \mathbb{N} \to \mathbb{R}$ is negligible if it approaches zero faster than every polynomial.

Probability distributions will be denoted by calligraphic letters $\mathcal{X}$, and $x \leftarrow \mathcal{X}$ denotes an element x drawn according to $\mathcal{X}$. Uniform distribution over $\{0,1\}^n$ is denoted by U_n. The statistical distance between two probability distributions $\mathcal{X}$ and $\mathcal{Y}$ over the same support S is denoted by $\Delta(X, Y) = \frac{1}{2} \sum_{a \in S} |\Pr[X = a] - \Pr[Y = a]|$.

The *min-entropy* of a distribution $\mathcal{X}$ that is defined over a set Ω is defined as $\mathrm{H}_\infty(\mathcal{X}) = \min_{\omega \in \Omega} \lg (1/\Pr[\mathcal{X} = \omega])$. An *$\alpha$-source* is a distribution $\mathcal{X}$ with $\mathrm{H}_\infty(\mathcal{X}) \geq \alpha$, and the *min-entropy rate* of an α-source over the set $\{0,1\}^n$ is α/n. A distribution $\mathcal{X}$ is *ϵ-close* to an α-source if there exists an α-source $\mathcal{Y}$ such that $\Delta(\mathcal{X}, \mathcal{Y}) \leq \epsilon$. The *average conditional min-entropy* [DORS08] of a random variable $\mathcal{X}$ given $\mathcal{Y}$ is defined as: $\tilde{\mathrm{H}}_\infty(\mathcal{X}|\mathcal{Y}) := -\lg \left(\mathrm{E}_{y \leftarrow \mathcal{Y}} \left[2^{-\mathrm{H}_\infty(\mathcal{X}|\mathcal{Y}=y)} \right] \right)$.

A family of hash functions $\mathcal{H} = \{h_i : \{0,1\}^n \to R\}$ is said to universal if for every *distinct* x_1, x_2, $\Pr_{h \leftarrow \mathcal{H}}[h(x_1) = h(x_2)] \leq 1/|R|$. A stronger notion is pairwise independence (p.i.): $\mathcal{H}$ is pairwise independent if for all distinct $x_1, x_2 \in \{0,1\}^n$ and all $y_1, y_2 \in R$, $\Pr_{h \leftarrow \mathcal{H}}[h(x_1) = y_1 \wedge h(x_2) = y_2] \leq 1/|R|^2$. We will actually need p.i. *permutations*, denoted perm, which can be efficiently inverted to recover the input.

Leftover hash lemma (LHL). We use two extension of the classical LHL [ILL89, HILL99]. The first version, called generalized LHL, is its generalization to the case of average conditional min-entropy [DORS08]:

Lemma 1 (Generalized LHL [DORS08])**.** *Let $\mathcal{X}, \mathcal{Y}$ be random variable such that $\mathcal{X} \in \{0,1\}^n$ and $\tilde{\mathrm{H}}_\infty(\mathcal{X}|\mathcal{Y}) \geq \alpha$. Let $\mathcal{H}$ be a family of universal hash functions. Then, for $h \leftarrow \mathcal{H}$, we have that: $\Delta\big(\ (\mathcal{Y}, h, h(\mathcal{X})),\ \ (\mathcal{Y}, h, \mathcal{U}_m)\ \big) \ \leq\ \epsilon$, provided $\alpha \geq m + 2\lg(1/\epsilon)$ and m is the output length of h.*

Classical LHL is obtained by dropping $\mathcal{Y}$ and replacing $\tilde{\mathrm{H}}_\infty(\mathcal{X}|\mathcal{Y})$ with $\mathrm{H}_\infty(\mathcal{X})$ in the above.

The second version we use is a further extension of the above lemma to the "crooked" case: roughly speaking, it says that if f is function with small range, and h is a pairwise independent permutation then $f(h(\mathcal{X}))$ is statistically independent of $\mathcal{X}$.[4]

Lemma 2 (Generalized "Crooked" LHL [DS05, BFO08])**.** *Let $\mathcal{X}, \mathcal{Y}$ be random variable such that $\mathcal{X} \in \{0,1\}^n$ and $\tilde{\mathrm{H}}_\infty(\mathcal{X}|\mathcal{Y}) \geq \alpha$. Let $\mathcal{H}$ be a family of pairwise independent functions from $\{0,1\}^n$ to a set R, and $f : R \to S$ be an arbitrary function to a set S. Then, for $h \leftarrow \mathcal{H}$, $\Delta\big(\ (\mathcal{Y}, h, f(h(\mathcal{X}))),\ \ (\mathcal{Y}, h, f(\mathcal{U}))\ \big)\ \ \leq\ \epsilon$, provided $\alpha \geq \log|S| + 2\lg(1/\epsilon)$ and (U) is the uniform distribution over the range R.*

Observe that h does not necessarily have a smaller range, and can be injective. Furthermore, the lemma holds for all functions f, even those, which potentially depend on $\mathcal{Y}$. The following lemma for average conditional min-entropy [DORS08] will be useful in the proofs.

Lemma 3. *If $\mathcal{Y}$ has 2^r values and $\mathcal{Z}$ is any random variable, then $\tilde{\mathrm{H}}_\infty(\mathcal{X}|(\mathcal{Y},\mathcal{Z})) \geq \mathrm{H}_\infty(\mathcal{X}|\mathcal{Y}) - r$.*

New notation. We will use two types of matrices: matrices of scalars, denoted by uppercase letters $A = [a_{ij}]$ and matrices of *vectors*, denoted by bold uppercase letters $\mathbf{A} = [\mathbf{a}_{ij}]$. When we want to be explicit about the dimension of a matrix $\mathbf{A}$, we will write $\mathbf{A}_{m\times n}$; define $A_{m\times n}$ similarly. Let $\mathbf{A}_{m\times n} = [\mathbf{a}_{ij}]$ be a matrix of vectors, and $\mathbf{x} = (x_1, \ldots, x_n)^\mathsf{T}$ be a column vector (with scalar coordinates), then we define:

$$\mathbf{A}\mathbf{x} = \mathbf{Y}_{m\times 1} = (\mathbf{y}_1, \ldots, \mathbf{y}_m)^\mathsf{T} \quad \text{where} \quad \mathbf{y}_i = \sum_{j\in[n]} x_j \mathbf{a}_{ij}$$

for every $i \in [m]$. That is, each $\mathbf{y}_i$ is a linear combination of the vectors in the i-th row of matrix $\mathbf{A}$, where the scalars of the linear combination are the coordinates of vector $\mathbf{x}$.

Let $\mathbb{G}$ be a group of prime order p, and g be its generator. For $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{Z}_p^n$, we define $g^{\mathbf{x}} := (g^{x_1}, \ldots, g^{x_n})$; similarly, we define matrices $g^A := [g^{a_{ij}}]$ and $g^{\mathbf{A}} := [g^{\mathbf{a}_{ij}}]$ where A is a matrix of scalars $a_{ij} \in \mathbb{Z}_p$, and $\mathbf{A}$ is a matrix of vectors $\mathbf{a}_{ij} \in \mathbb{Z}_p^\ell$ (for some $\ell \in \mathbb{N}$) for all values of i and j.

Finally, when dealing with vectors $\mathbf{x} = (x_1, \ldots, x_n)$ and $\mathbf{y} = (y_1, \ldots, y_n)$ both in $\mathbb{Z}_p^n$, we denote by $+$, $-$, and $\cdot$ the *component-wise* addition, subtraction, and multiplication modulo p. In particular, $\mathbf{x} \cdot \mathbf{y} = (x_1 y_1, \ldots, x_n y_n)$ with product taken modulo p. However, when we deal with vectors $\mathbf{u} = (u_1, \ldots, u_n)$ and $\mathbf{v} = (v_1, \ldots, v_n)$ both in $\mathbb{G}^n$ for some group $\mathbb{G}$, we use $\cdot$ to denote the group operation of $\mathbb{G}$ and define $\mathbf{u}\cdot\mathbf{v} = (u_1 \cdot v_1, \ldots, u_n \cdot v_n)$ as the component-wise group operation.

Bilinear groups and the matrix DDH assumption. We work with multiplicative groups $\mathbb{G}, \mathbb{G}_T$ of prime order p, equipped with a non-degenerate bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ satisfying the following property: for every $(a, b) \in \mathbb{Z}_p^2$ and generator $g \in \mathbb{G}$, $e(g^a, g^b) = e(g, g)^{ab}$. We require that the group operation and the map e are efficiently computable. We will assume an efficient

[4]Dodis and Smith [DS05] formulated and proved the crooked version of the classical LHL. The extension to the general case which works with average conditional min-entropy and any (lossy) function f was proved by Boldyreva et al. [BFO08].

generation algorithm $\mathcal{G}$, which on input a security parameter λ, outputs the description of groups $\mathbb{G}, \mathbb{G}_T$ of order p, the map e, and a generator g where p is a prime number of length $O(\lambda)$.

The "matrix DDH" assumption [BBS04, NS09], also known as the 3-linear assumption,[5] states that it is hard to tell whether a randomly chosen matrix $A \in \mathbb{Z}_p^{n\times 3}$ has rank 2 or 3 when given only g^A where $g \in \mathbb{G}$ is the generator of a prime-oder bilinear group $\mathbb{G}$. Formally, for every polynomial $n := n(\lambda)$, we have that distributions

$$\mathcal{D}_2(\lambda, n) \equiv \left\{\left(\mathsf{aux}, g^A\right) : A \overset{\$}{\leftarrow} \mathsf{Rank}_2^{n\times 3}(\mathbb{Z}_p)\right\}_{\lambda,n} \text{ and } \mathcal{D}_3(\lambda, n) \equiv \left\{\left(\mathsf{aux}, g^A\right) : A \overset{\$}{\leftarrow} \mathsf{Rank}_3^{n\times 3}(\mathbb{Z}_p)\right\}_{\lambda,n}$$

are computationally indistinguishable, where $\mathsf{aux} := (p, \mathbb{G}, \mathbb{G}_T, g, e) \leftarrow \mathcal{G}(\lambda)$ and $\lambda \in \mathbb{N}$. It follows from this assumption [NS09] that for all polynomially bounded $n, \ell \geq 3, r \geq 2, t \geq 0$, a random rank-$r$ matrix is computationally indistinguishable from a random rank-$(r + t)$ matrix when given in the exponent.

Random subspaces are leakage-resilient [BKKV10]. Brakerski, Kalai, Katz, and Vaikuntanathan [BKKV10] prove the following theorem which serves as as important tool in building their continual leakage resilient PKE scheme. The theorem roughly states that random subspaces of $\mathbb{Z}_p^n$ are "leakage resilient" provided that the leakage is "small" and independent of the subspace. More formally, let $X \subseteq \mathbb{Z}_p^n$ be a random subspace, $\mathbf{x_1}, \mathbf{x_2}$ two random vectors in X and $\mathbf{u_1}, \mathbf{u_2}$ two random vectors from the entire space. Let $\mathcal{L} : \mathbb{Z}_p^n \to W$ be a leakage function independent of X. The theorem states that if leakage is bounded (i.e. $|W|$ is small), then $\mathcal{L}(\mathbf{x})$ is *statistically close* to $\mathcal{L}(\mathbf{u})$ *even* given the subspace X.

Theorem 4 ([BKKV10])**.** *Let $n, \ell \in \mathbb{N}$, $n \geq \ell \geq 4$ and p be a prime. Let $X \overset{\$}{\leftarrow} \mathbb{Z}_p^{n\times\ell}$, $T \overset{\$}{\leftarrow} \mathsf{Rank}_2^{\ell\times 2}(\mathbb{Z}_p)$ and let $Y \overset{\$}{\leftarrow} \mathbb{Z}_p^{n\times 2}$. Let $\mathcal{L} : \mathbb{Z}_p^{n\times 2} \to W$ be some function independent of X such that $|W| \leq p^{\ell-3} \cdot \epsilon^2$ for a constant $\epsilon \in (0, 1)$. Then, $\Delta\big((X, \mathcal{L}(X \cdot T),\ (X, \mathcal{L}(Y))\big) \leq \epsilon$.*

BKKV encryption scheme. Our result uses the BKKV encryption which works as follows. The key generation algorithm chooses two uniformly random ℓ-dimensional vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ with elements from $\mathbb{Z}_p$, and another vector $\boldsymbol{t}$ uniformly at random from the orthogonal subspace of $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$. The public key is set to be $(g^{\boldsymbol{v}_1}, g^{\boldsymbol{v}_2})$ and the secret key is $g^{\boldsymbol{t}}$, where g is a generator of a bilinear pairing group $\mathbb{G}$ and exponentiation happens component-wise: $g^{\boldsymbol{t}} = (g^{t_1}, g^{t_2}, \ldots, g^{t_\ell})^{\mathsf{T}} \in \mathbb{G}^\ell$. To encrypt the bit 0, the encryption algorithm outputs a linear combination of $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ in the exponent: $g^{c_1\boldsymbol{v}_1 + c_2\boldsymbol{v}_2}$. The encryption of 1 is a completely random vector in the exponent. Notice that encryptions of 0 have exponents orthogonal to the secret key, while exponents of encryptions of 1 are not orthogonal to the secret key with overwhelming probability. During decryption, the respective components from the secret key and the ciphertext are paired via the bilinear pairing and the results are multiplied. As a result, if the vectors in the exponent are orthogonal, the result is the identity element. Otherwise, it is a random group element.

An important property of this encryption scheme is that it is almost *additively homomorphic*. It is easily shown that multiplying component-wise an encryption of $b_1 \in \{0, 1\}$ with an encryption of $b_2 \in \{0, 1\}$ provides an encryption of $b_1 + b_2$, except when $b_1 = b_2 = 1$ where you get an encryption of 1.

[5]We use the therms "matrix DDH assumption" and "linear assumption" interchangeably throughout the paper.

3 Lossy TDF under Continual Leakage

3.1 Our model

As noted earlier, we will define a "branch" based version of lossy trapdoor functions. To evaluate the function, an evaluator must specify a branch in addition to the input to the function. Some branches will be "lossy" whereas most other will be injective. The set of branches will be denoted $\mathcal{B}_\lambda$ for security parameter λ.

This primitive is closer to the all-but-one (ABO) primitive of Peikert and Waters [PW08]; the only difference is that in our formulation more than one branch might be lossy. In fact, there may exist branches which are neither injective nor "sufficiently lossy." Nevertheless, fraction of such branches will be negligible, and a random branch will be injective with overwhelming probability.

The leakage-resilience is formulated by requiring that it is hard to distinguish lossy branches from injective *even* under a continual leakage attack on the trapdoor. We will use a parameter $\rho \in [0, 1)$ to capture the leakage as the *fraction* of the length the secret-key. That is, in any given time period, the adversary can learn any PPT leakage function of the trapdoor t with output length at most $\rho|t|$. We now present the formal definition.

Lossy trapdoor functions resilient to continual memory leakage. Let $\lambda \in \mathbb{N}$ be the security parameter, and $n := n(\lambda)$ and $k := k(\lambda)$ be polynomials in λ. Parameter n denotes the length of the input to the function(s) and k is the *lossiness* parameter. For convenience, we define the *residual* information parameter $r := r(\lambda) = n(\lambda) - k(\lambda)$. Let $\rho := \{0,1\}^{\text{poly}(\lambda)} \to [0, 1)$ be a *leakage tolerance* parameter, representing the length of the leakage function as a fraction of the length the trapdoor. Finally, let $\{\mathcal{B}_\lambda\}_\lambda \in \mathbb{N}$ be the ensemble of branch-spaces, where $\mathcal{B}_\lambda = \{0,1\}^{\text{poly}(\lambda)}$.[6]

A collection of (n, k, ρ)-*continual leakage resilient lossy trapdoor functions* (CLR-LTDF) with the domain $\{0,1\}^n$ and branch collection $\mathcal{B} = \{\mathcal{B}_\lambda\}$ is given by a tuple of four (possibly probabilistic) polynomial-time (in λ) algorithms $(\mathsf{Setup}, \mathsf{Eval}, \mathsf{Inv}, \mathsf{Update})$ with the following specifications:

$\mathsf{Setup}(1^\lambda, b^*)$ takes as input 1^λ and a branch $b^* \in \mathcal{B}_\lambda$, and outputs (pp, t) where pp is a function index and t is its trapdoor.

$\mathsf{Eval}(\mathsf{pp}, b, x)$ is a deterministic algorithm which, given pp, a branch $b \in \mathcal{B}_\lambda$, and an input $x \in \{0,1\}^n$, outputs a value y.

$\mathsf{Update}(\mathsf{pp}, t)$ is a randomized algorithm which, given pp, and a trapdoor t for pp, outputs a new trapdoor t' such that $|t'| = |t|$. We call t' to be the *refreshed* or *updated* trapdoor.

$\mathsf{Inv}(\mathsf{pp}, t^*, y)$ is a deterministic algorithm which given pp, a trapdoor t^*, and a value y, outputs either $x \in \{0,1\}^n$ or $\perp$. Usually, either $t^* = t$ or t^* is obtained by repeated (at most polynomial) application of $\mathsf{Update}(\mathsf{pp}, \cdot)$.

We require that the following correctness and lossiness requirements hold.

1. *Injective, and invertible, on almost all branches:* except for a negligible fraction of $b \in \mathcal{B}_\lambda \setminus \{b^*\}$, algorithm $\mathsf{Eval}(\mathsf{pp}, b, \cdot)$ computes a (deterministic) injective function, which can be inverted using either the trapdoor t or *any of its polynomially-many refreshings.* That

[6]We can also consider more structured sets instead of $\{0,1\}^{\text{poly}(\lambda)}$. For example, a very intuitive and convenient choice is $\mathcal{B}_\lambda = \mathbb{Z}_p^m$; i.e., the branches vectors in $\mathbb{Z}_p^m$ for some $m = \text{poly}(n)$ and p is a prime of length λ. However, too much structure in $\mathcal{B}_\lambda$ should be avoided to ensure non-triviality and usefulness of the primitive.

is, for every polynomial $q := q(\lambda)$, every sufficiently large λ, every $x \in \{0,1\}^n$, and every $b^* \in \mathcal{B}_\lambda$,

$$\Pr\left[\mathsf{Inv}\left(\mathsf{pp}, t_q, \mathsf{Eval}(\mathsf{pp}, b, x)\right) \neq x \middle| \mathcal{E}\right] \leq \mu(\lambda),$$

where $\mathcal{E}$ is the event that $(\mathsf{pp}, t) \leftarrow \mathsf{Setup}(1^\lambda, b^*), b \leftarrow \mathcal{B}_\lambda, t_q$ is obtained by repeatedly applying the function $\mathsf{Update}(\mathsf{pp}, \cdot)$ to its own output (with fresh randomness) q times, starting from the initial input t and μ is a negligible function.[7]

2. *Lossy at the* given *branch* b^**:* for every $b^* \in \mathcal{B}_\lambda$, except with negligible probability over the randomness of Setup, we have: $|\mathsf{Eval}(\mathsf{pp}, b^*, \cdot)| \leq 2^{n-k}$, where $(\mathsf{pp}, t) \leftarrow \mathsf{Setup}(1^\lambda, b^*)$.

Finally, we require the following *hardness* properties.

1. *Indistinguishability of lossy branch under continual memory leakage:* We require that for every PPT algorithm A, it holds that

$$\left\{\mathbf{Game}_A^\rho(1^\lambda, 0)\right\}_{\lambda \in \mathbb{N}} \quad \equiv_c \quad \left\{\mathbf{Game}_A^\rho(1^\lambda, 1)\right\}_{\lambda \in \mathbb{N}} \tag{1}$$

where the variable $\mathbf{Game}_A^\rho(1^\lambda, d)$ is defined for $d \in \{0,1\}$ as follows:

$\mathbf{Game}_A^\rho(1^\lambda, d)$: The game proceeds between a challenger and adversary A in following stages:

(a) Init: The challenger chooses two branches (b, b^*) uniformly from the set $\mathcal{B}_\lambda$ and samples $(\mathsf{pp}, t) \leftarrow \mathsf{Setup}(1^\lambda, b^*)$. It sends pp to A.

(b) Leakage queries: A sends polynomially many leakage queries (in the form polynomial-sized circuits) $\mathcal{L}_1, \ldots, \mathcal{L}_s$ where $\mathcal{L}_i$ has output length at most $\rho|t|$, and $i \in [s]$ for some $s = \text{poly}(\lambda)$. The queries are chosen *adaptively* and answered as follows. At the start of this phase, the challenger sets $i = 1$ and $t_1 = t$. Upon receiving a leakage function $\mathcal{L}_i$ for $i \in [s]$, the challenger computes $\sigma_i = \mathcal{L}_i(t_i)$, $t_{i+1} \leftarrow \mathsf{Update}(\mathsf{pp}, t_i)$, and increases the counter i to $i+1$. It then returns σ_i to A and waits for the next leakage function.

(c) Challenge: Finally, if $d = 0$ the challenger sends b, and if $d = 1$ it sends b^* to the adversary.

The output of the game is whatever A outputs: w.l.o.g. the output is a *single* bit.

2. *Hard to sample a non-injective branch even given the inversion trapdoor:* roughly speaking, we require that no PPT algorithm A, given (pp, t) sampled by $\mathsf{Setup}(1^\lambda, b^*)$ for a *random* b^*, can compute a branch b such that the function $\mathsf{Eval}(\mathsf{pp}, b, \cdot)$ is *not* injective except with negligible probability. Formally, for every PPT algorithm A and every sufficiently large $\lambda \in \mathbb{N}$,

$$\Pr\left[b \in \mathcal{B}_\lambda \wedge \ |\mathsf{Eval}(\mathsf{pp}, b, \{0,1\}^n)| \neq 2^n \middle| \mathcal{E}\right] \leq \mu(\lambda),$$

where $\mathcal{E}$ denotes the event $b^* \leftarrow \mathcal{B}_\lambda$, $(\mathsf{pp}, t) \leftarrow \mathsf{Setup}(1^\lambda, b^*)$, $b \leftarrow A(\mathsf{pp}, t)$ and μ is a negligible function.

It is straightforward to extend this definition to allow leakage during setup and update phases.

[7]We note that this formulation does ensure that $\mathsf{Eval}(\mathsf{pp}, b, \cdot)$ is indeed an injective function for all but a negligible fraction of (pp, b) since inversion must almost always succeed for *every* given x.

Allowing general sampling algorithms. Our current formulation works with two uniform and independent branches (b, b^*) without worrying about *how* they are sampled. We consider a slightly more general definition where the algorithm for sampling the branches, say *Samp*, outputs an *encoded* branch (instead of the actual branch); a public decoding function is then used to compute the actual branch from the encoding. Two independent executions of *Samp* are used to sample branch encodings, which are then decoded. We require that the distribution of the decoded branches be statistically close to uniform over $\mathcal{B}_\lambda \times \mathcal{B}_\lambda$. In addition, during the CHALLENGE phase, the challenger sends the *encoding* of the challenge branch.

Formally, in the general definition, an (n, k, ρ)-CLR-LTDF is defined as above, except that we modify $\mathbf{Game}_A^\rho(1^\lambda, d)$ as follows.[8] For a non-uniform polynomial time algorithm *Samp* (with advice z) and a collection of decoding functions $\mathcal{H}$, we change the INIT phase as follows. The challenger samples a random decoding function $h \leftarrow \mathcal{H}$. It then samples two encodings $e \leftarrow \mathit{Samp}(1^\lambda, z)$ and $e^* \leftarrow \mathit{Samp}(1^\lambda, z)$ to define the branches $b = h(e)$ and $b^* = h(e^*)$. Branch b^* is used as the lossy branch (as before). We require that for a randomly chosen h, $(h(e), h(e^*))$ is statistically close to the uniform distribution over the pair of branches, and call $(\mathit{Samp}, \mathcal{H})$ to be *good* if they satisfy this requirement. Both h and pp are sent during INIT phase. During the CHALLENGE phase, if $d = 0$, encoding e is sent, otherwise e^* is sent. We require that (1) holds for all good $(\mathit{Samp}, \mathcal{H})$.

3.2 Our construction

We now present our construction. As described in Section 1.2, our construction of CLR-LTDF is inspired by the ABO functions of [PW08] which samples a $n \times n$ matrix $\mathbf{A}$ whose entries are (ElGamal) encryptions of 0 with correlated randomness. The function description is then set to $g^{\mathbf{M}}$ where $\mathbf{M} = \mathbf{A} + b^*\mathbf{I}$, $\mathbf{I}$ denotes the identity matrix, and b^* is lossy branch. The function evaluation on $(b, \mathbf{x})$ is $g^{(\mathbf{M} - b\mathbf{I})\mathbf{x}}$.

Our construction uses a similar approach and uses BKKV encryption instead of ElGamal. The branches in our construction will be a collection of n vectors, denoted by a matrix $\mathbf{B}_{n\times 1} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)^\mathsf{T}$ where every $\mathbf{b}_i \in \mathbb{Z}_p^\ell$. Each $\mathbf{b}_i$ can be interpreted as a BKKV encryption of 1, and it can also be used as a one-time pad for the diagonal elements of $\mathbf{A}$. Matrix $\mathbf{M}$ will now be matrix $\mathbf{A}$ whose diagonal entries are "masked", i.e., set to $\mathbf{a}_{ii} + \mathbf{b}_i$ for $i \in [n]$.

The construction. We will be working with prime-order bilinear groups. It will be convenient to assume the existence of a universal setup algorithm $\mathcal{G}(\lambda)$ which sets up some universal parameters such as the bilinear groups, bilinear map, a generator, and the set of branches $\mathcal{B}_\lambda$. That is, $\mathcal{G}(1^\lambda)$ is a randomized algorithm which outputs global parameters $\mathsf{params} = (p, \mathbb{G}, \mathbb{G}_T, g, e)$ where p is a random prime of length λ, $\mathbb{G}$ and $\mathbb{G}_T$ groups of order p, and g is a generators of $\mathbb{G}$. Let n, k, ρ be functions of λ, as defined earlier.

We define the set of branches to be $\mathcal{B}_\lambda = \{\mathbb{Z}_p^\ell\}^n$. That is, a branch $\mathbf{B} \in \mathcal{B}_\lambda$ is a collection of vectors (denoted as a matrix as per our notation) as: $\mathbf{B}_{n\times 1} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)^\mathsf{T}$ where $\mathbf{b}_i \in \mathbb{Z}_p^\ell$ for $i \in [n]$. We will assume that all algorithms described below has access to the global parameters params.[9] The four algorithms of our CLR-LTDF $\Pi_{\text{clr-ltdf}} := (\mathsf{Setup}, \mathsf{Eval}, \mathsf{Inv}, \mathsf{Update})$ are as follows.[10]

[8]We abuse the notation and continue to denote this modified game by $\mathbf{Game}_A^\rho$.

[9]We note that assuming such a $\mathcal{G}$ is only for convenience and without loss of generality. Indeed, we can assume $\mathcal{G}$ to be a part of the Setup algorithm. Since the length of the generated prime p is independent of p and only depends on λ, we can set $\mathcal{B}_\lambda = \left((\{0,1\}^{\lfloor \lg p \rfloor})^\ell\right)^n$ which is independent of p and always a subset of $(\mathbb{Z}_p^\ell)^n$.

[10]We remind the reader that uppercase letters, such as A, R, S, denote matrices of scalars (e.g., elements of $\mathbb{Z}_p$), whereas bold uppercase letters, such as $\mathbf{A}$, denote matrices of *vectors* (e.g. elements of $\mathbb{Z}_p^\ell$ or $\mathbb{G}^\ell$). Bold lowercase letter such as $\mathbf{x}$ represent vectors with only scalar entries.

$\mathsf{Setup}(1^\lambda, \mathbf{B}^*)$. Sample two uniformly random vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ in $\mathbb{Z}_p^\ell$, and two uniformly random $n \times n$ matrices of rank one, namely $R = [r_{ij}]$ and $S = [s_{ij}]$ from $\mathsf{Rank}_1^{n\times n}(\mathbb{Z}_p)$. Let $\mathbf{M} = [\mathbf{m}_{ij}]$ be a $n \times n$ matrix so that $\mathbf{m}_{ij} = r_{ij}\mathbf{v}_1 + s_{ij}\mathbf{v}_2$ for every valid i, j. The cell-entries of $\mathbf{M}$ are therefore vectors in $\mathsf{span}(\mathbf{v}_1, \mathbf{v}_2)$.

Let $\mathbf{B}^* = (\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*)^\mathsf{T} \in (\mathbb{Z}_p^\ell)^n$. Compute the matrix $\mathbf{A} := \mathbf{M} \boxplus \mathbf{B}^* := [\mathbf{a}_{ij}]$ as follows:

$$\mathbf{a}_{ij} = \begin{cases} \mathbf{m}_{ij} + \mathbf{b}_i^* & i = j \\ \mathbf{m}_{ij} & i \neq j \end{cases}$$

Note that the operation $\boxplus$ affects only the diagonal entries of $\mathbf{M}$. The public-parameter is defined to be $\mathsf{pp} = g^{\mathbf{A}}$. To compute the trapdoor, choose a matrix $T \overset{\$}{\leftarrow} \mathsf{kernel}^2(\mathbf{v}_1, \mathbf{v}_2)$; that is, T has two rows each of which is a vector in $\mathsf{kernel}(\mathbf{v}_1, \mathbf{v}_2)$. The trapdoor is set to $g^T \in \mathbb{G}^{2\times\ell}$. Output (pp, g^T).

$\mathsf{Eval}(\mathsf{pp}, \mathbf{B}, \mathbf{x})$. Let $\mathbf{x} \in \{0,1\}^n$ be a bit vector. The algorithm outputs $g^{\mathbf{Y}}$ such that:

$$g^{\mathbf{Y}} := (g^{\mathbf{y}_1}, \ldots, g^{\mathbf{y}_n})^\mathsf{T} = g^{\mathbf{A}'\mathbf{x}} \quad \text{where} \quad \mathbf{A}' = \mathbf{A} \boxplus (-\mathbf{B}).$$

Note that $g^{\mathbf{A}'}$ is easily computed given $(g^{\mathbf{A}}, \mathbf{B})$; likewise, $g^{\mathbf{Y}}$ is easily computed given $(g^{\mathbf{A}'}, \mathbf{x})$.[11]

$\mathsf{Update}(\mathsf{pp}, g^T)$. Choose a full rank matrix $V \in \mathbb{Z}_p^{2\times 2}$ and output $T' = g^{VT}$. The update operation essentially samples two random vectors in the row-span of T.

$\mathsf{Inv}(\mathsf{pp}, g^T, g^{\mathbf{Y}})$. Let $g^{\mathbf{Y}} = (g^{\mathbf{y}_1}, \ldots, g^{\mathbf{y}_n})^\mathsf{T}$ and let the two rows of g^T be $g^{\mathbf{t}_1}$ and $g^{\mathbf{t}_2}$. Output a bit vector $\mathbf{z} = (z_1, \ldots, z_n)$ where, for every $i \in [n]$, bit $z_i = 0$ if $\mathbf{y}_i$ is orthogonal to both $\mathbf{t}_1$ and $\mathbf{t}_2$; otherwise $z_i = 1$. Recall that if $g^{\mathbf{y}} = (g^{y_1}, \ldots, g^{y_\ell}) \in \mathbb{G}^\ell$ and $g^{\mathbf{t}} = (g^{t_1}, \ldots, g^{t_\ell}) \in \mathbb{G}^\ell$ for $\mathbf{y} \in \{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$ and $\mathbf{t} \in \{\mathbf{t}_1, \mathbf{t}_2\}$, then $\mathbf{y}$ is orthogonal $\mathbf{t}$ if and only if $\prod_i e\left(g^{y_i}, g^{t_i}\right) = 1$.

This completes the description of our function. The main theorem of this section is stated below. As noted earlier, the proof makes use of the fact that branches act as a one-time pad to successfully program that a future BKKV challenge can be appropriately mapped to either an injective or a lossy branch while still allowing the reduction to answer leakage queries. To be precise, the reduction works directly with a "matrix DDH" challenger instead of BKKV challenges. This is because the proof requires several steps that are specific to lossy trapdoor functions; it is unclear whether a "semi automatic" reduction to BKKV exists.

Theorem 5. *Under the validity of the matrix DDH assumption, the tuple* $(\mathsf{Setup}, \mathsf{Eval}, \mathsf{Update}, \mathsf{Inv})$ *specifies a* (n, k, ρ)*-CLR-LTDF over the domain* $\{0,1\}^n$ *for every polynomial* n *where* $k = n - 2\lg p$ *and* $\rho = \frac{1}{2} - \frac{3+\gamma}{\ell}$ *for every* $\ell \geq 7, \gamma > 0$

Proof. Let us first verify the correctness and lossiness of our construction. For any given branch $\mathbf{B}$ and input $\mathbf{x}$, consider the matrix $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n)^\mathsf{T} = \mathbf{A}'\mathbf{x} = (\mathbf{M} \boxplus (\mathbf{B}^* - \mathbf{B}))\,\mathbf{x}$. Letting $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_n)$, and expanding, we see that:

$$\mathbf{y}_i = x_i(\mathbf{b}_i^* - \mathbf{b}_i) + \mathbf{M}\mathbf{x}[i] \tag{2}$$

where $i \in [n]$ and $\mathbf{M}\mathbf{x}[i]$ denotes the i-th row of $\mathbf{M}\mathbf{x}$ (which is a vector in $\mathsf{span}(\mathbf{v}_1, \mathbf{v}_2)$). By construction, if $x_i = 0$ then $\mathbf{y}_i$ is orthogonal to both (trapdoor) vectors $\{\mathbf{t}_1, \mathbf{t}_2\}$, and if $x_i = 1$ it is not *except* when $(\mathbf{b}_i^* - \mathbf{b}_i)$ is also orthogonal to $\{\mathbf{t}_1, \mathbf{t}_2\}$. But the later happens with exactly $1/p^2$

[11]Recall that i-th row of $\mathbf{A}'\mathbf{x}$ contains a vector in the span of the vectors in the i-th row of $\mathbf{A}'$. See section 2.

probability (since there are two vectors) for a random $\mathbf{B}$. It follows that Inv works correctly (for all refreshings of the trapdoor) and the function is injective on almost all $\mathbf{B}$.

To see that the function is lossy when $\mathbf{B} = \mathbf{B}^*$, observe that in this case $\mathbf{Y} = \mathbf{M}\mathbf{x}$. Recall that $\mathbf{M}$ was constructed using matrices R and S of rank 1. In particular, $\mathbf{M}\mathbf{x}[i] = (R\mathbf{x}[i]) \cdot \mathbf{v}_1 + (S\mathbf{x}[i]) \cdot \mathbf{v}_2$. The range of both $R\mathbf{x}$ and $S\mathbf{x}$ is of size at most p, and therefore the size of the range of $\mathbf{M}\mathbf{x}$ is at most p^2. The function is lossy with $k = n - 2\lg p$.

The following lemmata prove the security properties of the construction.

Lemma 6. *For every PPT adversary A^*, $\left\{\mathbf{Game}^{\rho}_{A^*}(1^\lambda, 0)\right\}_{\lambda\in\mathbb{N}} \equiv_c \left\{\mathbf{Game}^{\rho}_{A^*}(1^\lambda, 1)\right\}_{\lambda\in\mathbb{N}}$ provided that $\rho < \frac{1}{2} - \frac{3}{\ell}$ for every $\ell \geq 7$.*

Proof. We prove the claim by designing a set of hybrid experiments. For simplicity we will drop subscripts and define hybrid $H_0(d) = \mathbf{Game}^{\rho}_{A^*}(1^\lambda, d)$. Next consider the following two hybrids:

$H_1(d)$**:** Same as $H_0(d)$ except that matrices R and S (see Setup) are sampled to be random matrices (and therefore are unlikely to have rank 1).

$H_2(d)$**:** Same as $H_1(d)$ except that this experiment updates the existing trapdoor differently. That is, whenever Update algorithm should be executed, the challenger in this experiment samples two uniformly random vectors in the space $\mathsf{kernel}(\mathbf{v}_1, \mathbf{v}_2)$ and sets that as the new trapdoor.[12]

We claim that under the linear assumption, $H_0(d)$ and $H_1(d)$ are computationally indistinguishable. This is because both experiments can be executed in polynomial time given (g^R, g^S) and without knowing their rank. Further, if R, S are rank-1 then the resulting experiment would be $H_0(d)$ and if they are random (therefore statistically close to full rank), it will be $H_1(d)$.

Next we claim that $H_1(d)$ and $H_2(d)$ are also computationally indistinguishable. To prove this, consider a random basis $X \in \mathbb{Z}_p^{\ell\times(\ell-2)}$ of the space $\mathsf{kernel}(\mathbf{v}_1, \mathbf{v}_2)$. Note that such a basis can be computed efficiently in polynomial time given $(\mathbf{v}_1, \mathbf{v}_2)$. Suppose that we are given a challenge matrix in the exponent g^C such that $C \in \mathbb{Z}_p^{(\ell-2)\times(\ell-2)}$ is either a random rank-2 matrix or a random full rank matrix. Further, if we let $g^{T_0} = g^T, g^{T_1}, \ldots$ represent the trapdoors after each update procedure, then we can set $g^{T_i} = g^{X.C.V_i}$ where $V_i \in \mathbb{Z}_p^{\ell-2\times 2}$ is random matrix; and this value can be efficiently computed given only g^C, X, V_i for every i. In addition, if C has rank 2 (resp., full) then the trapdoors are distributed identically to how they are distributed in $H_1(d)$ (resp., $H_2(d)$). Distinguish H_1 from H_2 therefore violates the linear assumption and the claim follows.[13]

To complete the proof, we need to show that,

$$H_2(0) \quad \equiv_c \quad H_2(1).$$

We prove this by setting up n hybrids $H_{2,1}, \ldots, H_{2,n}$. In these hybrids, instead of sending either $\mathbf{B}$ or $\mathbf{B}^*$ during the challenge phase, we will send a "hybrid" branch whose first i coordinates match $\mathbf{B}$ and rest $n - i$ match $\mathbf{B}^*$. Since these branches "mask" the diagonal of matrix $\mathbf{A}$, these hybrids are equivalent to setting up the diagonal of $\mathbf{A}$ so that first i entries, when "unmasked", reveal a BKKV-encryption of 0 and the rest $n - i$ that of 1.

Formally, for $i \in \{1, \ldots, n\}$ hybrid $H_{2,i}$ is defined below; define $H_{2,0}$ to be the same as $H_2(0)$.

[12] This is done easily since the challenger runs the Setup algorithm and therefore knows $(\mathbf{v}_1, \mathbf{v}_2)$.

[13] This part of the argument requires using two vectors in the trapdoor. If we use only one vector, we must assume (stronger) SXDH assumption. Under this assumption further optimizations and better parameters are possible.

$H_{2,i}$: This hybrid is identical to $H_{2,i-1}$ except that during the challenge phase it sends the "hybrid" branch $\mathbf{B}_i$ defined as follows. Let $\mathbf{B} = (\mathbf{b}_1, \dots, \mathbf{b}_n)$ and $\mathbf{B}^* = (\mathbf{b}^*_1, \dots, \mathbf{b}^*_n)$ be the branches sampled (by the challenger) at the start of the experiment. Then,

$$\mathbf{B}_i := (\mathbf{b}^*_1, \dots, \mathbf{b}^*_i, \mathbf{b}_{i+1}, \dots, \mathbf{b}_n).$$

Note that $H_{2,n}$ is the same as $H_2(1)$ since $\mathbf{B}_n = \mathbf{B}^*$.

To complete the proof, we need to show that $H_{2,i} \equiv_c H_{2,i+1}$ for every $i \in \{0, \dots, n-1\}$. This is the final part of the argument where the proof flows along the lines of BKKV with one key difference. As in BKKV, our challenger will also set up a "crooked" trapdoor from which it can answer leakage queries directly. However, in order to be able to apply theorem 4, it must ensure that prior to selecting the leakage function, the adversary has no information about the subspace from which the crooked trapdoor is sampled. This, however, is ensured by our construction since the branch $\mathbf{B}^*$ acts as a one-time pad to information theoretically hide the "BKKV challenge" that gets revealed in the end.

Formally, consider the following hybrid. (For convenience let us call $\mathbf{v}_1, \mathbf{v}_2$ in our procedures to be the *reference vectors* and $\mathbf{M}$ the *reference matrix*).

$H^*_{2,i}$: This hybrid uses a slightly different way to sample the parameters and trapdoors than $H_{2,i}$; nevertheless, it results in identical distributions. The hybrid uses a matrix $C \in \mathbb{Z}_p^{3\times 3}$ of rank *at least 2*, sampled uniformly at the start of the execution.[14]

The challenger samples a random matrix $Y \in \mathbb{Z}_p^{3\times \ell}$; it also samples a random basis $X \in \mathbb{Z}_p^{\ell\times(\ell-3)}$ for the space $\mathsf{kernel}(Y)$. We view Y as a row matrix and X as a column matrix. Next, the challenger computes the matrix CY and let $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ to be its 3 rows.

The challenger now uses vectors $\mathbf{v}_1, \mathbf{v}_2$ as reference vectors and samples $\mathbf{M}, \mathbf{B}^*, \mathbf{A}$ and $\mathbf{B}$ exactly as in the previous hybrid $H_{2,i}$. However, it sets $T \overset{\$}{\leftarrow} \mathsf{kernel}^2(\mathbf{v}_1, \mathbf{v}_2)$. Note that $\mathsf{span}(X)$ is always a subspace of $\mathsf{kernel}(\mathbf{v}_1, \mathbf{v}_2)$. Recall that the hybrid branch $\mathbf{B}_i$ is also set as before: first i coordinates from $\mathbf{B}^*$ and last $n-i$ from $\mathbf{B}$. Note that $\mathbf{v}_3$ is not used in this hybrid.

Since Y is random and C is a random matrix of rank≥ 2, distribution of $\mathbf{v}_1, \mathbf{v}_2$ is identical in $H_{2,i}$ and $H^*_{2,i}$. All other values are sampled in the same way in both hybrids, and therefore $H_{2,i}$ and $H^*_{2,i}$ are identical.

Next, we intend to use "crooked" trapdoors and apply theorem 4. However, the application of this theorem is slightly tricky and we actually need to consider two cases separately: depending on whether rank of C is 3 or 2. This makes the description of the next hybrid look somewhat complex, but actually it is simple at a high level. There are two main changes. First, the trapdoors come from the subspace defined by $\mathsf{span}(\mathbf{X})$. Second, we wish to output a branch $\mathbf{B}_{ch}$ so that $\mathbf{a}_{jj} - \mathbf{B}_{ch}[j] = \mathbf{v}_3$ where $j = i+1$ is for succinctness. The purpose of the second part will become clear shortly. The description follows.

$G_i^{(2)}$: The challenger samples $C \leftarrow \mathbb{Z}_p^{3\times 3}$ of rank 2. It also samples matrices Y and X as before, and sets $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$ to be the rows of matrix CY. It then samples $\mathbf{B}, \mathbf{B}^*$ as before and a random challenge vector $\mathbf{c} \in \mathbb{Z}_p^{\ell}$. Finally, the remaining quantities are sampled as follows (for succinctness we let $j = i+1$):

[14] Jumping ahead, eventually g^C will be received from an external challenger of the matrix DDH assumption.

1. Sample all entries of matrix $\mathbf{A}$ as before except $\mathbf{a}_{jj}$, which is set to $\mathbf{a}_{jj} = \mathbf{v}_3 + \mathbf{c}$.
2. Sample the trapdoor $T \xleftarrow{\$} \mathsf{span}^2(\mathbf{X})$; all subsequent updates are also sampled in the same way.
3. In the challenge phase, send the following *challenge* branch:
$$\mathbf{B}_{ch} := (\mathbf{b}_1^*, \ldots, \mathbf{b}_i^*, \mathbf{c}, \mathbf{b}_{i+2}, \ldots, \mathbf{b}_n)$$
where $\mathbf{b}_{i'} = \mathbf{B}[i'], \mathbf{b}_{i'}^* = \mathbf{B}^*[i']$ for $i' \in [n]$.

$G_i^{(3)}$**:** Same as above except that C has full rank (i.e., three).

We first prove that $G_i^{(2)}$ is statistically close to $H_{2,i+1}^*$. The proof is as follows. First we note that $\mathbf{A}$ is distributed identically in both hybrids since $\mathbf{a}_{jj}$ is uniformly random in $G_i^{(2)}$ as well. The only difference is that the leakage occurs on vectors from $\mathsf{span}(\mathbf{X})$ which is a subspace of $\mathsf{kernel}(\mathbf{v}_1, \mathbf{v}_2)$. Note that $g^{\mathbf{A}}$ contains no information about the subspace $\mathbf{X}$.(In fact this holds even if $\mathbf{v}_3$ were given, since C has rank 2). Therefore we can apply theorem 4 to conclude that the execution of the two hybrids are statistically close up to the challenge phase. When $\mathbf{B}_{ch}$ is revealed, note that $\mathbf{c}$ is revealed. The distribution of $\mathbf{c}$ is uniform under the condition that $\mathbf{a}_{jj} - \mathbf{c}$ distributed uniformly in $\mathsf{span}(\mathbf{v}_1, \mathbf{v}_2)$. Note that this is identical to the distribution of $\mathbf{b}_{i+1}^*$ in game $H_{2,i+1}$; all other branch components are identically distributed in both hybrids. To compute the statistical distance between $G_i^{(2)}$ and $H_{2,i+1}^*$, note that the random subspace X has dimension $l \times l - 3$. Let W be the range of any leakage function $\mathcal{L}$. Since $\mathcal{L}$ leaks ρ fraction of the secret key, $|W| = p^{\rho.2l}$. From theorem 4 it follows that the statistical distance is bounded by $\text{poly}(\lambda)\sqrt{|W|/p^{l-6}}$. For this quantity to be negligible, it suffices to satisfy $\rho.2l < l - 6$. Hence, for all $\gamma > 0$, $\rho = \frac{l-6-\gamma}{2l}$ ensures that $H_{2,i+1}$ and $G_i^{(2)}$ are statistically indistinguishable.

Next we prove that $G_i^{(3)}$ is statistically close to $H_{2,i}^*$. By the same argument as above $g^{\mathbf{A}}$ is distributed identically in both hybrids. In addition, since $\mathbf{c}$ is chosen independently and uniformly, distribution of $\mathbf{v}_3$ is independent from $g^{\mathbf{A}}$. Therefore, the leakage function is also independent of $\mathbf{v}_3$ and hence the subspace $\mathbf{X}$. We can therefore apply theorem 4 as above to conclude that the execution of these hybrids are statistically close up to the challenge phase. When $\mathbf{B}_{ch}$ is revealed, note that $\mathbf{c}$ is revealed. The distribution of $\mathbf{c}$ is uniform under the condition that $\mathbf{a}_{jj} - \mathbf{c} = \mathbf{v}_3$ is uniformly distributed in $\mathbb{Z}_p^\ell$. Note that this is identical to the distribution of $\mathbf{b}_{i+1}$ in game $H_{2,i}$; all other branch components are identically distributed in both hybrids. Note that revealing $\mathbf{v}_3$ is equivalent to revealing the subspace $\mathbf{X}$. However, since theorem 4 holds even given $\mathbf{X}$, we have that the two hybrids remain statistically close even after the challenge phase.

Finally, observe that both hybrids $G_i^{(2)}$ and $G_i^{(3)}$ can be executed perfectly even if instead of sampling C internally, they receive g^C from an outside party. It follows from the linear assumption that $G_i^{(2)} \equiv_c G_i^{(3)}$. Consequently, $H_{2,i}$ and $H_{2,i+1}$ are also computationally indistinguishable. This completes the proof of the lemma. □

Lemma 7. *It is hard to sample a non-injective branch against our scheme.*

Proof. Suppose that there exists a PPT adversary A^* which outputs a non-injective branch with probability $\epsilon = 1/q$ for some polynomial q. That is, A returns a branch containing a coordinate $\mathbf{b}_i$ for some $i \in [n]$ such that $\mathbf{b}_i^* - \mathbf{b}_i$ is orthogonal to both trapdoor vectors $\mathbf{t}_1, \mathbf{t}_2$. We show how to use A^* to solve the discrete log problem.

Consider the following adversary $\mathcal{B}$ which, except with negligible probability, on input a vector $g^{\mathbf{b}^*} \in \mathbb{G}^\ell$ returns a random vector $\mathbf{t}$ and a scalar α such that $\langle \mathbf{t}, \mathbf{b}_i^* \rangle = \alpha$ (all computations modulo some prime p). To do so, $\mathcal{B}$ first runs the setup algorithm our construction on a random $\mathbf{B}^*$ and let $\mathbf{A} = [\mathbf{a}_{ij}]$ be the matrix in the exponent. $\mathcal{B}$ multiplies $g^{\mathbf{b}^*}$ to the j-th element of the diagonal so that the new matrix in the exponent becomes $\mathbf{A}'$ which is same $\mathbf{A}$ except that $\mathbf{a}'_{jj} = \mathbf{a}_{jj} + \mathbf{b}^*$. Trapdoors for $\mathbf{A}$ are also valid trapdoors for $\mathbf{A}'$ and distributed uniformly; except that now the lossy branch becomes $\mathbf{B}^{*'}$—same as $\mathbf{B}^*$ but j-th coordinate differ by an additive term $\mathbf{b}^*$. $\mathcal{B}$ runs A^* on these new parameters (distributed correctly) to obtain a $\mathbf{b}^*_i$ so that $(\mathbf{a}'_{ii} - \mathbf{b}^*_i)$ is orthogonal to the trapdoor vectors and $j = i$ with probability $\epsilon/n = 1/nq$.[15] Repeating this experiment $n^2/\epsilon = n^2 q$ times finds a $\mathbf{b}^*_j$ with probability $\exp(-n)$. If $\mathbf{b}^*_j$ is found in one of the experiments, $\mathcal{B}$ returns one of the trapdoors in that experiment, say $\mathbf{t}$, and its scalar product $\alpha = \langle \mathbf{t}, (\mathbf{b}^*_j - \mathbf{a}_{jj}) \rangle$ where $\mathbf{a}_{jj}$ is j-th diagonal element of $\mathbf{A}$.

Running $\mathcal{B}$ for ℓ times yields ℓ equations with entries of $\mathbf{b}^*$ as variables; these equations are linearly independent w.h. probability. Solving the equations yieldss $\mathbf{b}^*$. The discrete-log challenge is solved by planting it in one of the coordinates of $\mathbf{b}^*$ and sampling the rest uniformly. This completes proof. □

□

An interesting property of our functions is that they are actually universal on the lossy branches. I.e., they have a low collision probability. We prove this extra feature in Section A.

Remark. Our proof is not sensitive to *how* the branches are sampled in the game, so long as they are uniform and independent over the branch space. Therefore, it actually proves the general version of the definition where the branch encodings are sampled using an arbitrary non-uniform PPT sampler *Samp*, and then decoded efficiently using a random function from $\mathcal{H}$.

4 Leakage Resilient Deterministic PKE

In this section, we define leakage-resilient D-PKE and its security, and show that our CLR-LTDF yield such a scheme when branch b is set to $h(m)$ provided certain conditions on h and the entropy of m are met.

4.1 Modeling deterministic PKE under continual leakage

A deterministic public-key encryption scheme is a triple of polynomial-time algorithms $\Pi = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Update})$. The key-generation algorithm KG is a randomized algorithm which on input the security parameter 1^λ outputs a pair (pk, sk) of a public key pk and a secret key sk. The encryption algorithm Enc is *deterministic*, takes as input 1^λ, a public key pk, and a plaintext $m \in \{0,1\}^{n(\lambda)}$, and outputs a ciphertext $c \in \{0,1\}^{t(\lambda)}$. The (possibly deterministic) decryption algorithm Dec takes as input 1^λ, a secret key sk, and a ciphertext $c \in \{0,1\}^{t(\lambda)}$, and outputs either a plaintext $m \in \{0,1\}^{n(\lambda)}$ or the special symbol $\perp$.

Algorithm $\mathsf{Update}(pk, sk')$ is a randomized update algorithm which, given pk, and a secret-key sk' for pk, outputs a new secret-key sk'' such that $|sk''| = |sk'|$; input sk' is either sk or one of the outputs of Update. We call sk' to be the *refreshed* or *updated* secret-key. We require that the outputs of Dec is identical on sk or sk' (which is output of Update). For succinctness, we will always assume 1^λ as an implicit input to all algorithms and refrain from explicitly specifying it.

We now define our security notion, namely CLR-PRIV1-IND security. It is essentially a direct extension of the standard PRIV1-IND definition in the leakage-free setting. PRIV1-IND is the

[15] Orthogonality is tested using pairings and therefore knowing $g^{\mathbf{a}'_{jj}}$ and the trapdoor is sufficient.

"single challenge" version of the PRIV-IND definition which requires that, roughly speaking, the encryptions of two sequences of messages be computationally indistinguishable provided that each message has sufficient min-entropy and the sequences have same "equality pattern." Formally, these requirements are captured by defining a *α-source q-message adversary*, which is also relevant to our definition.[16]

The α-source q-message adversary. Let $A = (A_1, A_2)$ be a probabilistic polynomial-time algorithm, and let $\alpha = \alpha(\lambda)$ and $q = q(\lambda)$ be functions of the security parameter $\lambda \in \mathbb{N}$. For any $\lambda \in \mathbb{N}$ denote by $(\mathcal{M}_\lambda^{(0)}, \mathcal{M}_\lambda^{(1)}, \mathcal{STATE}_\lambda)$ the distribution corresponding to the output of $A_1(1^\lambda)$. Then, A is a *α-source q-message adversary* if the following properties hold:

1. $\mathcal{M}_\lambda^{(b)} = \left(\mathcal{M}_{1,\lambda}^{(b)}, \ldots, \mathcal{M}_{q,\lambda}^{(b)}\right)$ is a distribution over sequences of q plaintexts for each $b \in \{0,1\}$.
2. For any $\lambda \in \mathbb{N}$, $i, j \in [q]$, and for every triplet $\left(\left(m_1^{(0)}, \ldots, m_q^{(0)}\right), \left(m_1^{(1)}, \ldots, m_q^{(1)}\right), \mathsf{state}\right)$ that is produced by $A_1(1^\lambda)$ it holds that $m_i^{(0)} = m_j^{(0)}$ if and only if $m_i^{(1)} = m_j^{(1)}$.
3. For any $\lambda \in \mathbb{N}$, $b \in \{0,1\}$, $i \in [q]$, and $\mathsf{state} \in \{0,1\}^*$ it holds that $\mathcal{M}_{i,\lambda}^{(b)}|_{\mathcal{STATE}_\lambda = \mathsf{state}}$ is a $\alpha(\lambda)$-source.

We are now ready to define the continual-leakage-resilient version of PRIV1-IND security, namely CLR-PRIV1-IND. The definition is same as PRIV1-IND security except that the adversary is allowed to ask leakage queries *before* receiving the challenge encryption.

CLR-PRIV1-IND Security. A deterministic public-key encryption scheme $\Pi = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Update})$ is *CLR-PRIV1-IND-secure for $\alpha(\lambda)$-source 1-message adversaries* with leakage-parameter ρ if for any probabilistic polynomial-time $\alpha(\lambda)$-source 1-message adversary $A = (A_1, A_2)$ there exists a negligible function $\nu(\lambda)$ such that

$$\mathbf{Adv}_{\Pi,A,\lambda}^{\mathsf{CLR-PRIV1-IND}} \overset{\mathsf{def}}{=} \left| \begin{array}{l} \Pr\left[\mathsf{Expt}_{\Pi,A,\lambda}^{\mathsf{CLR-PRIV1-IND}}(0) = 1\right] \\ -\Pr\left[\mathsf{Expt}_{\Pi,A,\lambda}^{\mathsf{CLR-PRIV1-IND}}(1) = 1\right] \end{array} \right| \leq \nu(\lambda)$$

for all sufficiently large $\lambda \in \mathbb{N}$, where $\mathsf{Expt}_{\Pi,A,\lambda}^{\mathsf{CLR-PRIV1-IND}}(b)$ is defined as follows:

1. Init: sample $(pk, sk) \leftarrow \mathsf{KG}(1^\lambda)$, and send pk to A_2.
2. Leakage queries: set $i = 1$ and $sk_1 = sk$; interact with A_2, answering every leakage query $\mathcal{L}_i$ whose output length is at most $\rho \cdot |sk|$, as follows. Send $\sigma_i = \mathcal{L}_i(sk_i)$ to A_2, and set $sk_{i+1} \leftarrow \mathsf{Update}(pk, sk_i)$ and $i = i + 1$.
3. Challenge: sample $(m_0, m_1, \mathsf{state}) \leftarrow A_1(1^\lambda)$ and set $c \leftarrow \mathsf{Enc}_{pk}(m_b)$; send (c, state) to A_2. Output of A_2 is the output of the experiment.

Remark. It is not necessary to sample $(m_0, m_1, \mathsf{state}) \leftarrow A_1(1^\lambda)$ in the Challenge phase. Instead, they can be sampled during the Init phase so long as they are kept completely outside the view of the adversary; Challenge phase then only computes c and sends (c, state) to A_2. From here on, we shall work with this modified version. Also, the advantage can also be mentioned in terms of probability p of correctly guessing which distribution the encrypted message comes from; it is easy to see that the advantage above comes out to be $|2p - 1|$.

[16]We will only focus on the single challenge setting; it is straightforward to extend our definition to deal with sequence of messages and get the corresponding notion CLR-PRIV-IND. However, our construction only satisfies the single message definition, and we do not know if our scheme can be shown to satisfy security for multiple messages.

4.2 Our deterministic public-key encryption scheme

The construction. Let $\Pi_{\text{clr-ltdf}} := (\mathsf{Setup}, \mathsf{Eval}, \mathsf{Inv}, \mathsf{Update})$ be a (n, k, ρ)-CLR-LTDF which satisfies the universal hash property with respect to lossy branches. Let $s = s(\lambda)$ be a polynomial describing the length of branches $b \in \mathcal{B}_\lambda$.[17] Let $\mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}^s\}$ be a family of universal hash functions, and perm be a family of pairwise independent permutations which are easy to invert.[18] The message space of the scheme is $\{0,1\}^n$.

The key-generation algorithm KG samples $h \leftarrow \mathcal{H}$, $\pi \leftarrow \mathsf{perm}$, $b^* \leftarrow \mathcal{B}_\lambda$, $(\mathsf{pp}, t) \leftarrow \mathsf{Setup}(1^\lambda, b^*)$. It outputs $pk = (h, \pi, \mathsf{pp})$ and $sk = t$. The encryption algorithm Enc takes as input pp and a message $x \in \{0,1\}^n$. It outputs $y = \mathsf{Eval}(\mathsf{pp}, b, \pi(x))$ where $b = h(x)$ is used as the branch. The decryption algorithm Dec takes as input y, sk, and $pk = (h, \pi, \mathsf{pp})$. It outputs $x = \pi^{-1}(\mathsf{Inv}(\mathsf{pp}, sk, y))$ if x is valid and $\perp$ otherwise. The update algorithm of the scheme is the same as Update.

We denote this scheme by $\Pi_{\text{clr-de}} := (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Update})$. It is easy to verify the correctness of this scheme.

Theorem 8. *Scheme $\Pi_{\text{clr-de}}$ is a CLR-PRIV1-IND secure scheme for $\alpha(\lambda)$-source 1-message adversary with leakage parameter ρ provided that $\Pi_{\text{clr-ltdf}}$ is a (n, k, ρ)-CLR-LTDF and: $\alpha(\lambda) \geq n(\lambda) - k(\lambda) + s(\lambda) + 2\lg(1/\epsilon)$, where ϵ is an arbitrary negligible function in λ.*

Proof. The overview for the proof is as follows. We will show that the advantage of every α-source 1-message adversary $A = (A_1, A_2)$ is at most $2\delta + 6\epsilon$ where δ is distinguishing advantage for the CLR-LTDF functions and ϵ is the statistical gap from the (generalized crooked) LHL.

To do so, we will start by designing five hybrid games, $H_1, \ldots, H_5$ where H_1 is essentially the experiment $\mathsf{Expt}^{\mathsf{CLR-PRIV1-IND}}_{\Pi_{\text{clr-de}}, A, \lambda}(0)$. Hybrid H_5 will be independent of the any underlying distribution imposed by the adversary on the messages. Analogously, we write hybrids $H_5, \ldots, H_9$ (in the reverse order) where we achieve the same conclusion for the second experiment $\mathsf{Expt}^{\mathsf{CLR-PRIV1-IND}}_{\Pi_{\text{clr-de}}, A, \lambda}(1)$.

In order to reduce the security of our scheme to that of CLR-LTDF, we will create a situation in our hybrids such that the branch corresponding to *only the challenge message* becomes lossy. This is rather tricky since, unlike previous usage of lossy trapdoor functions for deterministic encryption, we cannot simply switch to the lossy mode or a pre-selected lossy branch. This also requires that we work with the generalized version of CLR-LTDF where the challenger sends *encoded* branches.

We now describe the hybrid experiments. Let $A = (A_1, A_2)$ be our α-source 1-message adversary.

H_1**:** This hybrid experiment is identical to the experiment $\mathsf{Expt}^{\mathsf{CLR-PRIV1-IND}}_{\Pi_{\text{clr-de}}, A, \lambda}(0)$. In particular, the experiment first samples $(m_0, m_1, \mathsf{state}) \leftarrow A_1(1^\lambda)$, and then (independently) samples $h \leftarrow \mathcal{H}$, $\pi \leftarrow \mathsf{perm}$, $b^* \leftarrow \mathcal{B}_\lambda$, and $(\mathsf{pp}, t) \leftarrow \mathsf{Setup}(1^\lambda, b^*)$, to define $pk = (h, \pi, \mathsf{pp})$ and $sk = t$. It gives pk to A_2 and answers the leakage queries repeatedly while updating sk everytime. In the end, it sends the ciphertext $c = \mathsf{Eval}(\mathsf{pp}, b, \pi(m_0))$ to A_2, where $b = h(m_0)$ is used as the branch. The output of A_2 is the output of the experiment.

H_2**:** This hybrid is identical to H_1 except that instead of sampling b^* randomly from $\mathcal{B}_\lambda$, it uses the following specific method to generate it: samples an independent triplet $(m'_0, m'_1, \mathsf{state}') \leftarrow A_1(1^\lambda)$ and computes $b^* = h(m'_0)$.

We observe that since m'_0 is sampled independently, and has min-entropy at least α, by the (standard) LHL b^* is ϵ-close to uniform distribution over $\mathcal{B}_\lambda$. Consequently, the statistical distance between (the outputs of) H_2 and H_1 is at most ϵ.

[17] W.l.o.g. we can assume s to be quite small if necessary. If the length requires a large string to describe the branch, we can use pseudorandom generators of sufficient stretch.

[18] Such permutations are known.

H_3**:** This hybrid is identical to H_2 except that instead of encrypting m_0 in the last phase, it encrypts m'_0. I.e., the ciphertext is $c = \mathsf{Eval}(\mathsf{pp}, b', \pi(m'_0))$ where $b' = h(m'_0)$. Observe that, $b' = b^*$ since they are both $h(m'_0)$.

Let δ_2 denote the advantage in distinguishing the outputs of H_2 and H_3. Let δ denote the advantage in distinguishing the lossy and injective branches of $\Pi_{\mathrm{clr-de}}$. From lemma 9, $\delta_2 = \delta$.

H_4**:** Identical to H_3 except that the value of $\pi(m'_0)$ is replaced by a uniformly random value U (from the domain) to compute c. I.e., it computes c as $c = \mathsf{Eval}(\mathsf{pp}, b', U)$. Note that b' is still equal to $h(m'_0)$ so that $b' = b^*$ as before.

From lemma 10, H_3 and H_4 are ϵ-close.

H_5**:** Identical to H_4 except that the branch b^* is sampled uniformly at random. That is, the hybrid does not set $b^* = h(m'_0)$ any more, and just samples it randomly.

Since m'_0 has large min-entropy α, and h is independent of m'_0, by the standard LHL, we have that H_4 and H_5 are ϵ-close.

In the rest of the proof, we now "reverse the hybrids" using the *second component* of A_1's output so that we eventually reach the second experiment, namely $\mathsf{Expt}^{\mathsf{CLR-PRIV1-IND}}_{\Pi_{\mathrm{clr-de}},A,\lambda}(1)$.

H_6**:** Same as H_5 except that instead of sampling b^* uniformly, it is computed as $b^* = h(m'_1)$. Recall that m'_1 was sampled from an independent run of A_1, and $c = \mathsf{Eval}(\mathsf{pp}, b^*, U)$ in this hybrid.

As before, from the standard LHL, H_6 and H_5 are ϵ-close.

H_7**:** Same as H_6 except that the uniform value U is replaced by $\pi(m'_1)$. I.e., as in the previous hybrid, public parameters pp are sampled using $b^* = h(m'_1)$, but c is computed as $c = \mathsf{Eval}(\mathsf{pp}, b^*, \pi(m'_1))$.

H_7 and H_6 are ϵ-close; the proof is identical to that of lemma 10.

H_8**:** Same as H_7 except that c is set to be an encryption of m_1 instead of m'_1. I.e., as before $b^* = h(m'_1)$ and it is used to sample pp. However, $c = \mathsf{Eval}(\mathsf{pp}, b, \pi(m_1))$ where $b = h(m_1)$.

H_7 and H_8 can be distinguished with advantage at most δ. The proof is identical to that of lemma 9 except that instead of using the sampler $Samp_0$, it uses the following sampler $Samp_1(1^\lambda)$: (1) sample $(m_0, m_1, \mathsf{state}) \leftarrow A_1(1^\lambda)$; (2) output $e = (m_1, \mathsf{state})$ as the encoded branch.[19]

H_9**:** Same as H_8 except that the branch b^* is sampled uniformly at random and values $(m'_0, m'_1, \mathsf{state}')$ are not sampled any more. H_8 and H_9 are ϵ-close due to standard LHL.

H_9 is essentially the experiment $\mathsf{Expt}^{\mathsf{CLR-PRIV1-IND}}_{\Pi_{\mathrm{clr-de}},A,\lambda}(1)$. It follows that A's advantage in breaking our scheme is at most $2\delta + 6\epsilon$ which is negligible.

□

Lemma 9. $\delta_2 = \delta$.

Proof. We prove the lemma by constructing an adversary B against $\Pi_{\mathrm{clr-de}}$. We work with the general security definition for indistinguishability of lossy branches.

[19]The decoding function remains the same: $h(e) = h(m, \mathsf{state}) = h(m)$, but since $m = m_1$ for $Samp_1$, we will have $b = h(m_1)$.

Recall that in this definition, the adversary specifies a sampling algorithm for generating *encoded* branches, and the challenger selects a good decoding function to compute the branches from these encodings. Therefore, before describing B, we first describe a sampling algorithm, $Samp_0$ and corresponding decoding functions.

Sampler $Samp_0(1^\lambda)$**:**

1. Sample $(m_0, m_1, \mathsf{state}) \leftarrow A_1(1^\lambda)$.
2. Output $e := (m_0, \mathsf{state})$ as the encoded branch.

Decoding: The set of decoding functions is the family of universal hash functions $\mathcal{H}$; a random $h \leftarrow \mathcal{H}$ is fixed as the decoding function.[20] A branch b corresponding to an encoding $e = (m, \mathsf{state})$ is defined as: $b = h(e) = h(m, \mathsf{state}) := h(m)$.

Before B can use $Samp_0$, we must argue that $(Samp_0, \mathcal{H})$ are good, i.e., the branch $b = h(e)$ is statistically close to uniform distribution over the branch space. This, however, is trivially true due to the (standard) LHL since the first component of e, namely $m = m_0$, has min-entropy at least α and h is an independently chosen universal hash function.

We are now ready to describe B. Adversary B participates in the indistinguishability game for lossy/injective branches — namely $\mathbf{Game}_B^\rho$ — with a challenger. B internally uses the adversary $A = (A_1, A_2)$ who distinguishes between hybrids H_2 and H_3. B works as follows.

Adversary B**.** It uses algorithm A_1 to define the sampler $Samp_0$ as above and sends $Samp_0$ to the challenger. Upon receiving (h, pp) from the challenger, B samples $\pi \leftarrow \mathsf{perm}$ to define $pk = (h, \pi, \mathsf{pp})$. It sends pk to A_2. If A_2 asks any leakage queries, B forwards them to the challenger and returns its answers back to A_2. Finally, after all leakage queries are over, B receives an encoding $e = (m, \mathsf{state})$ from the challenger. B computes $c = \mathsf{Eval}(\mathsf{pp}, h(m), \pi(m))$ and sends (c, state) to A_2. B outputs whatever A_2 outputs.

We argue that if the challenger sends an encoding e corresponding to an injective branch, then B perfectly simulates hybrid H_2 for A; otherwise it perfectly simulates H_3. The challenger in $\mathbf{Game}_B^\rho$ starts by sampling a uniform h and two independent branch encodings $e = (m_0, state_0)$ and $e^* = (m_0^*, state_0^*)$ using the sampler $Samp_0$. It then sets $b = h(m_0)$ as the injective branch, $b^* = h(m_0^*)$ as the lossy branch, and samples $(\mathsf{pp}, t) \leftarrow \mathsf{Setup}(1^\lambda, b^*)$. It then sends (h, pp) to B who samples π honestly. Therefore the distribution of pk, as well as the answer to the leakage queries are perfectly simulated by B for A (identical in H_2 and H_3). If the challenger sends the injective encoding, then B receives $e = (m_0, \mathsf{state}_0)$, and therefore A receives $c = \mathsf{Eval}(\mathsf{pp}, h(m_0), \pi(m_0))$. This is identical to c in H_2. On the other hand, if the challenger sends the lossy encoding $e^* = (m_0^*, \mathsf{state}_0^*)$, A receives $c = \mathsf{Eval}(\mathsf{pp}, h(m_0^*), \pi(m_0^*))$, as in H_3. In both cases, the state variable is also appropriately distributed. This proves the lemma. □

Lemma 10. *H_3 and H_4 are ϵ-close.*

Proof. We invoke the generalized *crooked* LHL (lemma 2) to claim that H_3 and H_4 are statistically close. Let $\mathcal{M}_\lambda^{(0)}$ denote the random variable corresponding to the first and the last output of A_1; i.e., it is of the form (m_0, state).

Let us first observe that H_3 can be perfectly simulated given only the tuple $(\pi, h, b^*, \mathsf{pp}, \mathsf{state}, c)$ where $c = \mathsf{Eval}(\mathsf{pp}, b^*, \pi(m_0'))$ and pp is sampled using $\mathsf{Setup}(1^\lambda, b^*)$.

[20]Function h will be sampled by the challenger during the Init phase of the lossy/injective game.

Let $\mathcal{X} = \mathcal{M}_\lambda^{(0)}$ and $\mathcal{Y} = (h, h(\mathcal{X}), \mathsf{pp}, \mathsf{state})$ for $h \leftarrow \mathcal{H}$. Furthermore, for every $y \leftarrow \mathcal{Y}$ define the function f_y as follows (note that y is of the form $(h, b^*, \mathsf{pp}, \mathsf{state})$): $f_y(\cdot) = \mathsf{Eval}(\mathsf{pp}, b^*, \cdot)$. Note that f_y, for every y in the support of $\mathcal{Y}$ is a lossy function with range of size at most 2^{n-k}.

With this notation, we have that H_3 can be perfectly simulated given $(\pi, y, f_y(x))$ where $x \leftarrow \mathcal{X}$ and y is computed from x: i.e., $y = (h, h(x), \mathsf{pp}, \mathsf{state})$.

Now observe that π is independent of $\mathcal{X}$, the average conditional min-entropy of $\mathcal{X}|\mathcal{Y}$ is at least $n - k - 2\lg\epsilon$ (by our choice of α and lemma 3), and f_y has range of size at most 2^{n-k} (since it corresponds to the evaluation of Eval on the lossy branch b^*) for every value of y in the support of $\mathcal{Y}$. Therefore, by the generalized crooked LHL, the tuple is ϵ-close to the distribution $(\pi, y, f_y(\mathcal{U}))$ for every value of y in the support where $\mathcal{U}$ is the uniform distribution over the domain. (and hence over the entire support since it is a convex combination of individual distributions defined by each y); . It follows that the distributions $(\pi, \mathcal{Y}, f_\mathcal{Y}(\mathcal{X}))$ and $(\pi, \mathcal{Y}, f_\mathcal{Y}(\mathcal{U}))$ are ϵ-close where we denote by $f_\mathcal{Y}$ the distribution over functions f_y for $y \leftarrow \mathcal{Y}$. This is because the two distributions are ϵ-close "point-wise", i.e. for every "point" y.[21]

Finally, observe that the distribution $(\pi, \mathcal{Y}, f_\mathcal{Y}(\mathcal{U}))$ takes values of the form $(\pi, h, b^*, \mathsf{pp}, \mathsf{state}, c)$ where $c = \mathsf{Eval}(\mathsf{pp}, b^*, \mathcal{U})$, which is sufficient to perfectly simulate H_4. Therefore H_3 and H_4 are ϵ-close. □

References

[ADN+10] J. Alwen, Y. Dodis, M. Naor, G. Segev, S. Walfish, and D. Wichs. Public - key encryption in the bounded - retrieval model. In *EUROCRYPT*, pages 113–134, 2010.

[AGP14] Prabhanjan Ananth, Vipul Goyal, and Omkant Pandey. Interactive proofs under continual memory leakage. In *CRYPTO*, pages 164–182, 2014.

[AGV09] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In *TCC*, pages 474–495, 2009.

[AK97] Ross J. Anderson and Markus G. Kuhn. Low cost attacks on tamper resistant devices. In *Security Protocols Workshop*, pages 125–136, 1997.

[BBO07] Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. Deterministic and efficiently searchable encryption. In *Advances in Cryptology – CRYPTO '07*, pages 535–552, 2007.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *Advances in Cryptology – CRYPTO '04*, pages 41–55, 2004.

[BFO08] Alexandra Boldyreva, Serge Fehr, and Adam O'Neill. On notions of security for deterministic encryption, and efficient constructions without random oracles. In *Advances in Cryptology – CRYPTO '08*, pages 335–359, 2008.

[BFOR08] Mihir Bellare, Marc Fischlin, Adam O'Neill, and Thomas Ristenpart. Deterministic encryption: Definitional equivalences and constructions without random oracles. In *Advances in Cryptology – CRYPTO '08*, pages 360–378, 2008.

[21]Note that generalized crooked LHL (lemma 2) applies even though $f_\mathcal{Y}$ depends on $\mathcal{Y}$.

[BGJK12] Elette Boyle, Shafi Goldwasser, Abhishek Jain, and Yael Tauman Kalai. Multiparty computation secure against continual memory leakage. In *STOC*, pages 1235–1254, 2012.

[BKKV10] Zvika Brakerski, Yael Tauman Kalai, Jonathan Katz, and Vinod Vaikuntanathan. Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In *FOCS*, pages 501–510, 2010.

[BKPW12] Mihir Bellare, Eike Kiltz, Chris Peikert, and Brent Waters. Identity-based (lossy) trapdoor functions and applications. In *EUROCRYPT*, pages 228–245, 2012.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st Annual ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[BS11] Zvika Brakerski and Gil Segev. Better security for deterministic public-key encryption: The auxiliary-input setting. In *Advances in Cryptology – CRYPTO '11*, pages 543–560, 2011.

[BSW11] Elette Boyle, Gil Segev, and Daniel Wichs. Fully leakage-resilient signatures. In *EUROCRYPT*, pages 89–108, 2011.

[CDRW10] Sherman S. M. Chow, Yevgeniy Dodis, Yannis Rouselakis, and Brent Waters. Practical leakage-resilient identity-based encryption from simple assumptions. In *ACM Conference on Computer and Communications Security*, pages 152–161, 2010.

[DHLAW10] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Cryptography against continuous memory attacks. In *FOCS*, pages 511–520, 2010.

[DKL09] Yevgeniy Dodis, Yael Tauman Kalai, and Shachar Lovett. On cryptography with auxiliary input. In *STOC*, pages 621–630, 2009.

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

[DP08] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *FOCS*, pages 293–302, 2008.

[DS05] Yevgeniy Dodis and Adam Smith. Correcting errors without leaking partial information. In *STOC 2005*, pages 654–663, 2005.

[FGK$^+$10] David Mandell Freeman, Oded Goldreich, Eike Kiltz, Alon Rosen, and Gil Segev. More constructions of lossy and correlation-secure trapdoor functions. In *Proceedings of the 13th International Conference on Practice and Theory in Public Key Cryptography*, pages 279–295, 2010.

[FOR12] Benjamin Fuller, Adam O'Neill, and Leonid Reyzin. A unified approach to deterministic encryption: New constructions and a connection to computational entropy. Cryptology ePrint Archive, Report 2012/005, 2012.

[FRR$^+$10] Sebastian Faust, Tal Rabin, Leonid Reyzin, Eran Tromer, and Vinod Vaikuntanathan. Protecting circuits from leakage: the computationally-bounded and noisy cases. In *EUROCRYPT*, pages 135–156, 2010.

[GJS11] Sanjam Garg, Abhishek Jain, and Amit Sahai. Leakage resilient zero knowledge. In *Advances in Cryptology – CRYPTO*, 2011.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[GR12] Shafi Goldwasser and Guy N. Rothblum. How to compute in the presence of leakage. In *FOCS*, pages 31–40, 2012.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[HLOV11] Brett Hemenway, Benoît Libert, Rafail Ostrovsky, and Damien Vergnaud. Lossy encryption: Constructions from general assumptions and efficient selective opening chosen ciphertext security. In *ASIACRYPT*, pages 70–88, 2011.

[HLWW13] Carmit Hazay, Adriana López-Alt, Hoeteck Wee, and Daniel Wichs. Leakage-resilient cryptography from minimal assumptions. In *EUROCRYPT*, pages 160–176, 2013.

[Hof12] Dennis Hofheinz. All-but-many lossy trapdoor functions. In *EUROCRYPT*, pages 209–227, 2012.

[ILL89] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions (extended abstracts). In *STOC*, pages 12–24, 1989.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In *CRYPTO*, pages 463–481, 2003.

[Koc96] Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *CRYPTO*, pages 104–113, 1996.

[LLW11] Allison B. Lewko, Mark Lewko, and Brent Waters. How to leak on key updates. In *STOC*, 2011.

[LRW11] Allison B. Lewko, Yannis Rouselakis, and Brent Waters. Achieving leakage resilience through dual system encryption. In *TCC*, pages 70–88, 2011.

[LW10] Allison B. Lewko and Brent Waters. On the insecurity of parallel repetition for leakage resilience. In *FOCS*, pages 521–530, 2010.

[MPRS12] Ilya Mironov, Omkant Pandey, Omer Reingold, and Gil Segev. Incremental deterministic public-key encryption. In *EUROCRYPT*, pages 628–644, 2012.

[MR04] S. Micali and L. Reyzin. Physically observable cryptography. In *TCC*, pages 278–296, 2004.

[NS09] Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. In *CRYPTO*, 2009.

[O'N10] Adam O'Neill. Deterministic public-key encryption revisited. Eprint Report 2010/533, 2010.

[OST06] Dag Arne Osvik, Adi Shamir, and Eran Tromer. Cache attacks and countermeasures: The case of aes. In *CT-RSA*, pages 1–20, 2006.

[Pan14] Omkant Pandey. Achieving constant round leakage-resilient zero-knowledge. In *TCC*, 2014.

[PRZB11] Raluca A. Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan. Cryptdb: protecting confidentiality with encrypted query processing. In *SOSP*, pages 85–100, 2011.

[PRZB12] Raluca A. Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan. Cryptdb: processing queries on an encrypted database. *Commun. ACM*, 55(9):103–111, 2012.

[PW08] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In *STOC*, pages 187–196, 2008.

[QLCC13] Baodong Qin, Shengli Liu, Kefei Chen, and Manuel Charlemagne. Leakage-resilient lossy trapdoor functions and public-key encryption. In *AsiaPKC*, 2013.

[QS01] Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In *E-smart*, pages 200–210, 2001.

[RSV$^+$11] Mathieu Renauld, François-Xavier Standaert, Nicolas Veyrat-Charvillon, Dina Kamel, and Denis Flandre. A formal study of power variability issues and side-channel attacks for nanoscale devices. In *EUROCRYPT*, pages 109–128, 2011.

[RSV13] Ananth Raghunathan, Gil Segev, and Salil P. Vadhan. Deterministic public-key encryption for adaptively chosen plaintext distributions. In *EUROCRYPT*, 2013.

[Wee12] Hoeteck Wee. Dual projective hashing and its applications - lossy trapdoor functions and more. In *EUROCRYPT*, 2012.

A Universal Hash Property

We prove here that our construction satisfies the stronger requirement of universal hash property on lossy branches.

Lemma 11. *Let* $\Pi_{\mathrm{clr-ltdf}} := (\mathsf{Setup}, \mathsf{Eval}, \mathsf{Inv}, \mathsf{Update})$ *be the construction in 3.2. Then, for every sufficiently large* $\lambda \in \mathbb{N}$ *and every* $b^* \in \mathcal{B}_\lambda$*, the function family* $\{F_{\mathsf{pp},b^*}\}$ *over* $\{0,1\}^n$ *defined as* $F_{\mathsf{pp},b^*}(x) := \mathsf{Eval}(\mathsf{pp}, b^*, x)$ *is universal* with respect to the following algorithm *for sampling the functions: compute* $(\mathsf{pp}, t) \leftarrow \mathsf{Setup}(1^\lambda, b^*)$ *and output* (pp, b^*).

Proof. We show that for every distinct $\mathbf{x}_1, \mathbf{x}_2$, the collision probability $\Pr[F_{\mathsf{pp},b^*}(\mathbf{x}_1) = F_{\mathsf{pp},b^*}(\mathbf{x}_2)] \leq 1/\xi$; here $\xi \leq p^2$ is the size of the range of F_{pp,b^*} and the probability is taken over the sampling of pp using Setup. Recall that p is a large prime fixed by the global setup algorithm $\mathcal{G}$

Recall that sampling pp involves sampling a matrix $\mathbf{M}$, and from equation (2), $F_{\mathsf{pp},b^*}(\mathbf{x}) = \mathbf{M}\mathbf{x}$ where i-th entry is a vector $\mathbf{y}_i$ of the form:

$$\mathbf{y}_i := \mathbf{M}\mathbf{x}[i] = R\mathbf{x}[i] \cdot \mathbf{v}_1 + S\mathbf{x}[i] \cdot \mathbf{v}_2$$

for $i \in [n]$. Further recall that R and S are rank 1 matrices and $\mathbf{v}_1$ and $\mathbf{v}_2$ are random vectors.

Vectors $\mathbf{v}_1, \mathbf{v}_2$ are not parallel w.h.p.; it will be convenient to work with orthogonal vectors. Therefore, let $\mathbf{u}_1$ be the component of $\mathbf{v}_2$ that is parallel to $\mathbf{v}_1$, i.e., $\mathbf{u}_1 = \frac{\langle \mathbf{v}_1, \mathbf{v}_2 \rangle}{||\mathbf{v}_2||} \cdot \mathbf{v}_1$. Then $\mathbf{u}_2 = \mathbf{v}_2 - \mathbf{u_1}$ is orthogonal to $\mathbf{v}_1$ and we can write $\mathbf{y}_i$ in terms of $\mathbf{u}_1$ and $\mathbf{u}_2$:

$$\mathbf{y}_i = (R+S)\mathbf{x}[i] \cdot \mathbf{u}_1 + S\mathbf{x}[i] \cdot \mathbf{u}_2 \tag{3}$$

A collision occurs for two vectors $\mathbf{x}_1, \mathbf{x}_2$ if and only if $\mathbf{M}\mathbf{x}_1 = \mathbf{M}x_2 \Rightarrow \mathbf{M}(\mathbf{x}_1 - \mathbf{x}_2) = 0$. Using (3), it follows that since $\mathbf{u}_1 \perp \mathbf{u}_2$, this can happen if and only if $R(\mathbf{x}_1 - \mathbf{x}_2) = S(\mathbf{x}_1 - \mathbf{x}_2) = 0$. That is, if we define $h_A(\mathbf{x}) = A\mathbf{x}$ for a rank-1 $n \times n$ matrix in $\mathbb{Z}_p$ and $\mathbf{x} \in \{0,1\}^n$, then a collision occurs for F_{pp,b^*} if and only if collision occurs on both of its component functions h_R and h_S. It is easy to check that h_A is a universal hash function, and since R and S independently chosen, it follows that F_{pp,b^*} is also universal. $\square$

B Construction of CLR-LTDFs Based on SXDH Assumption

In this section, we describe a leakage resilient lossy trapdoor function family that allows optimal leakage bounds. This scheme is based on the symmetric external Diffie-Hellman (SXDH) assumption. Let $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ be groups of prime order p, and let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a non-degenerate bilinear map. The SXDH assumption states that the DDH problem is hard in both $\mathbb{G}_1$ and $\mathbb{G}_2$.

This construction is similar to the one in Section 3.2, with minor modifications in the Setup and Update algorithms. Eval and Init are the same as before.

$\mathsf{Setup}(1^\lambda, \mathbf{B}^*)$. Sample a uniformly random vector $\mathbf{v_1}$ from $\mathbb{Z}_p^l$, and a uniformly random $n \times n$ rank one matrix $R = [r_{ij}]$ from $\mathsf{Rank}_1^{n\times n}(\mathbb{Z}_p)$. Let $\mathbf{M} = [\mathbf{m_{ij}}]$ be an $n \times n$ matrix such that $\mathbf{m_{ij}} = r_{ij}\mathbf{v_1}$ for all $1 \le i, j \le n$.

Let $\mathbf{B}^* = (\mathbf{b_1^*}, \ldots, \mathbf{b_n^*})^{\mathsf{T}}$. Define matrix $\mathbf{A} := \mathbf{M} \boxplus \mathbf{B}^* := [\mathbf{a_{ij}}]$ as follows:

$$\mathbf{a_{ij}} = \begin{cases} \mathbf{m_{ij}} & i \neq j \\ \mathbf{m_{ij}} + \mathbf{b_i^*} & i = j \end{cases}$$

The public parameter is defined to be $\mathsf{pp} = g_1^{\mathbf{A}}$. To compute the trapdoor, choose $\mathbf{t} \xleftarrow{\$} \mathsf{kernel}(\mathbf{v_1})$. The trapdoor is set to be $g_2^{\mathbf{t}}$. Output $(\mathsf{pp}, g_2^{\mathbf{t}})$.

$\mathsf{Update}(\mathsf{pp}, g_2^{\mathbf{t}})$. To update the trapdoor $g_2^{\mathbf{t}}$, sample $r \xleftarrow{\$} \mathbb{Z}_p^l$ and output $g_2^{r\mathbf{t}}$.

Theorem 12. *Under the validity of the SXDH assumption, the tuple* (Setup, Eval, Inv, Update) *specifies a (n, k, ρ)-CLR-LTDF over the domain $\{0,1\}^n$ with $k = n - \lceil \log p \rceil$, and $\rho = 1 - \frac{3+\gamma}{l}$ for all $\gamma > 0$.*

Proof. The proof of correctness is similar to what we had for the original construction. To prove that the function is lossy when branch $\mathbf{B} = \mathbf{B}^*$, note that in this case, $\mathbf{Y} = \mathbf{M}\mathbf{x}$, where $\mathbf{M}\mathbf{x}[i] = R\mathbf{x}[i]\cdot\mathbf{v_1}$. $R\mathbf{x}$ can take p different values, and therefore $\mathbf{Y}$ can take p different values. The function is lossy with $k = n - \lceil \log p \rceil$.

The hybrids are essentially the same as before. The only place where we require the SXDH assumption is when we go from hybrid $H_1(d)$ to $H_2(d)$. Here, we use the SXDH assumption to prove that rank 1 and rank $l-3$ matrices in the exponent are computationally indistinguishable.

To compute the leakage bonds, note that in the SXDH case, the random subspace X has dimension $l \times l - 2$. Let W be the range of any leakage function $\mathcal{L}$. Since $\mathcal{L}$ leaks ρ fraction of the secret key, $|W| = p^{\rho \cdot l}$. From an analogue of Theorem 4 for single vectors, it follows that the statistical distance between $H_{2,i+1}$ and $G_i^{(2)}$ (and similarly the statistical distance between $H_{2,i}$ and $G_i^{(3)}$) is bounded by $poly(\lambda)\sqrt{|W|/p^{l-3}}$. For this quantity to be negligible, it suffices to satisfy $\rho.l < l - 3$. Hence, for all $\gamma > 0$, $\rho = \frac{l-3-\gamma}{l}$ ensures that $H_{2,i+1}$ and $G_i^{(2)}$ are statistically indistinguishable. □

C Application to CCA Security under Continual Leakage

In this section we demonstrate, by constructing a CCA secure PKE in the continual leakage model, that our formulation of CLR-LTDF is of general interest. It can be just as useful in "leakage world" as the original lossy TDFs in the "leakage free" setting.

We now describe how to construct a leakage-resilient CCA secure cryptosystem using our leakage-resilient LTDF primitive. Our construction is similar to the respective construction of [PW08] and the security proof follows closely their security proof. Our CCA secure encryption system uses four underlying primitives: a CLR-LTDF family, an ABO-LTDF family, a pairwise independent hash function family, and an one-time strongly unforgeable signature scheme. Notice that to achieve leakage resilience only the first family has to be leakage resilient. In [PW08] there was a regular LTDF instead of a CLR-LTDF. All the parameters of our system depend on the security parameter λ and therefore, for simplicity, we omit this explicit dependence.

Let $(\mathsf{Setup}_{\mathsf{clr}}, \mathsf{Eval}_{\mathsf{clr}}, \mathsf{Inv}_{\mathsf{clr}}, \mathsf{Update}_{\mathsf{clr}})$ be a collection of (n, k, ρ)-continual leakage resilient lossy trapdoor functions (CLR-LTDF) with branch sets $\mathcal{B}_\lambda = \{0,1\}^v$. Also, let $(\mathsf{Setup}_{\mathsf{abo}}, \mathsf{Eval}_{\mathsf{abo}}, \mathsf{Inv}_{\mathsf{abo}})$ be a collection of (n, k') all-but-one lossy trapdoor functions (ABO-LTDF) with branch sets $\mathcal{B}'_\lambda = \{0,1\}^w$, and let $(\mathsf{Setup}_\sigma, \mathsf{Sign}_\sigma, \mathsf{Vrfy}_\sigma)$ be an one-time strongly unforgeable scheme (OT-SUF) with verification keys $vk \in \{0,1\}^w = \mathcal{B}'_\lambda$. We require that the total lossiness $k + k' \geq n + \kappa$ for a parameter $\kappa = \omega(\log(n))$. Finally, let $\mathcal{H}$ be a pairwise independent hash function family from $\{0,1\}^n$ to $\{0,1\}^\ell$ where $\ell \leq \kappa - 2\log(1/\epsilon)$ for some negligible $\epsilon \in \mathsf{negl}(\lambda)$. Our cryptosystem Π consists of the following four polynomial time algorithms and its message space is $\{0,1\}^\ell$.

$\mathsf{Setup}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$: The algorithm first picks a uniformly random branch $b^* \xleftarrow{R} \{0,1\}^v = \mathcal{B}_\lambda$. Then it calls $\mathsf{Setup}_{\mathsf{clr}}(1^\lambda, b^*) \to (\mathsf{pp}_{\mathsf{clr}}, t)$ and $\mathsf{Setup}_{\mathsf{abo}}(1^\lambda, 0^w) \to (\mathsf{pp}_{\mathsf{abo}}, t^*)$. Also it picks a uniformly random hash function $h \xleftarrow{R} \mathcal{H}$. It outputs:

$$\mathsf{pk} = (\mathsf{pp}_{\mathsf{clr}}, \mathsf{pp}_{\mathsf{abo}}, h), \qquad \mathsf{sk} = t$$

Notice that the branch b^* and the trapdoor t^* are not used.

$\mathsf{Encrypt}(\mathsf{pk}, m) \to \mathsf{ct}$: The encryption algorithm first generates a verification - signing key pair from the one-time signature scheme: $\mathsf{Setup}_\sigma(1^\lambda) \to (\mathsf{vk}_\sigma, \mathsf{sk}_\sigma)$. It picks a uniformly random branch $b \xleftarrow{R} \{0,1\}^v = \mathcal{B}_\lambda$ and a uniformly random input for the trapdoor functions: $x \xleftarrow{R} \{0,1\}^n$. It computes:

$$c_0 = m \oplus h(x), \quad c_1 = \mathsf{Eval}_{\mathsf{clr}}(\mathsf{pp}_{\mathsf{clr}}, b, x), \quad c_2 = \mathsf{Eval}_{\mathsf{abo}}(\mathsf{pp}_{\mathsf{abo}}, \mathsf{vk}_\sigma, x), \quad c_3 = b$$

Finally, it signs the tuple (c_0, c_1, c_2, c_3) with $\mathsf{Sign}_\sigma(\mathsf{sk}_\sigma, (c_0, c_1, c_2, c_3)) \to \sigma$ and outputs

$$\mathsf{ct} = (c_0, c_1, c_2, c_3, \mathsf{vk}_\sigma, \sigma)$$

$\mathsf{Decrypt}(\mathsf{pk}, \mathsf{sk}, \mathsf{ct}) \to m$: The decryption algorithm gets $\mathsf{ct} = (c_0, c_1, c_2, c_3, \mathsf{vk}_\sigma, \sigma)$ and fails immediately by outputting $\perp$ if $\mathsf{Vrfy}_\sigma(\mathsf{vk}_\sigma, (c_0, c_1, c_2, c_3), \sigma) = \mathsf{false}$. Otherwise, it computes $x =$

$\mathsf{Inv_{clr}}(\mathsf{pp_{clr}}, t, c_1)$ and checks if $c_1 = \mathsf{Eval_{clr}}(\mathsf{pp_{clr}}, c_3, x)$ and $c_2 = \mathsf{Eval_{abo}}(\mathsf{pp_{abo}}, \mathsf{vk}_\sigma, x)$. If not the algorithm outputs $\perp$. Otherwise it outputs $m = c_0 \oplus h(x)$.

$\mathsf{Update}(\mathsf{pk}, \mathsf{sk}) \rightarrow \mathsf{sk}'$: The update algorithm takes a public key $\mathsf{pk} = (\mathsf{pp_{clr}}, \mathsf{pp_{abo}}, h)$ and a secret key $\mathsf{sk} = t$ and outputs $\mathsf{sk}' = t'$ where $t' = \mathsf{Update_{clr}}(\mathsf{pp_{clr}}, t)$.

We argue that the above system is a correct, continual-leakage resilient, CCA-secure public key encryption system. Firstly we will prove that Π decrypts correctly all messages with probability 1 minus negligible.

Let $\mathsf{ct} = (c_0, c_1, c_2, c_3, \mathsf{vk}_\sigma, \sigma)$ be a ciphertext produced by the above encryption algorithm. During decryption we would have that $\mathsf{Vrfy}_\sigma(\mathsf{vk}_\sigma, (c_0, c_1, c_2, c_3), \sigma) = \mathsf{true}$ according to the correctness of the one-time signature scheme. Therefore the decryption algorithm would compute $x = \mathsf{Inv_{clr}}(\mathsf{pp_{clr}}, t', c_1)$, where t' is the secret key that might have been updated several times. According to the injective property of CLR-LTDFs , this would return the original $x \in \{0,1\}^n$ unless with negligible probability $\mu(\lambda)$. If it returns the original x, both checks $c_1 = \mathsf{Eval_{clr}}(\mathsf{pp_{clr}}, c_3, x)$ and $c_2 = \mathsf{Eval_{abo}}(\mathsf{pp_{abo}}, \mathsf{vk}_\sigma, x)$ would succeed with probability 1, since both evaluation algorithms are deterministic. Since h is also deterministic, the decryption algorithm retrieves the original message m.

What remains is to prove the following theorem regarding the CCA security of Π.

Theorem 13. *Scheme* $\Pi = (\mathsf{Setup}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Update})$ *is a ρ-continual-leakage resilient CCA secure public key encryption system under the assumptions that* $(\mathsf{Setup_{clr}}, \mathsf{Eval_{clr}}, \mathsf{Inv_{clr}}, \mathsf{Update_{clr}})$ *is (n, k, ρ)-CLR-LTDF,* $(\mathsf{Setup_{abo}}, \mathsf{Eval_{abo}}, \mathsf{Inv_{abo}})$ *is (n, k')-ABO-LTDF,* $(\mathsf{Setup}_\sigma, \mathsf{Sign}_\sigma, \mathsf{Vrfy}_\sigma)$ *is an OT-SUF signature scheme, $\mathcal{H}$ is a pairwise independent hash function family, and all parameters satisfy the above relations.*

In order to prove theorem 13 we define the following sequence of games. For completeness, we briefly describe the continual leakage CCA game.

Continual Leakage CCA Game: In the Init phase the challenger calls $\mathsf{Setup}(1^\lambda)$ and gives the public key pk to the attacker. In query Phase-1, the attacker can make leakage queries and decryption queries to the challenger in a fully adaptive way. The leakage queries are of the form of polynomial-sized circuits $\mathcal{L}_i$ with output size at most $\rho|\mathsf{sk}|$. The challenger replies to the attacker with $\mathcal{L}_i(\mathsf{sk})$ and updates the secret key to $\mathsf{sk}' = \mathsf{Update}(\mathsf{pk}, \mathsf{sk})$. The decryption queries consist of ciphertexts ct_i and the challenger responds with $\mathsf{Decrypt}(\mathsf{pk}, \mathsf{sk}, \mathsf{ct}_i)$. In the challenge phase Challenge, the attacker sends two messages m_0 and m_1, the challenger flips a random coin $c \xleftarrow{R} \{0,1\}$, and it replies with the ciphertext $\mathsf{ct}^* = \mathsf{Encrypt}(\mathsf{pk}, m_c)$. The next phase is query Phase-2 where the attacker can only make decryption queries as above. The only restriction is that all the queried ciphertexts must be different than the challenge ciphertext ct^*. Finally, in the Guess phase the attacker tries to guess the bit c.

Game-1: The first game we define for our scheme is the same as the **CL-CCA** game with the difference that the verification key vk^*_σ that is used in the challenge ciphertext ct^* is generated at the beginning of the game (in the Init phase). Namely, in the Init phase the challenger calls $\mathsf{Setup}_\sigma(1^\lambda) \rightarrow (\mathsf{vk}^*_\sigma, \mathsf{sk}^*_\sigma)$ and in the Challenge phase the $\mathsf{Encrypt}(\mathsf{pk}, m_c)$ call uses $(\mathsf{vk}^*_\sigma, \mathsf{sk}^*_\sigma)$ to generate the ciphertext instead of generating a new key pair.

Game-2: This game is the same as **Game-1** except the treatment of the decryption queries. In this game if the attacker makes a decryption query $\mathsf{ct}_i = (c_0, c_1, c_2, c_3, \mathsf{vk}_\sigma, \sigma)$ with $\mathsf{vk} = \mathsf{vk}^*$, then the challenger immediately replies with $\perp$, instead of calling the decryption algorithm. Notice that since vk^* is generated and known to the challenger at the beginning of the game, this is a well-defined security game.

Game-3: This game differs from the previous one only in the INIT phase. Remember that the challenger calls $\mathsf{Setup}_\sigma(1^\lambda) \to (\mathsf{vk}^*_\sigma, \mathsf{sk}^*_\sigma)$ at the beginning of this phase. After doing that it calls a new setup algorithm for the scheme $\mathsf{Setup}^*(1^\lambda)$ which is the same as $\mathsf{Setup}(1^\lambda)$ except that the ABO-LTDF is created with $\mathsf{Setup}_{\mathsf{abo}}(1^\lambda, \mathsf{vk}^*) \to (\mathsf{pp}_{\mathsf{abo}}, t^*)$. Namely, the ABO lossy branch is $\mathsf{vk}^* \in \mathcal{B}'$.

Game-4: The only difference of this game from **Game-3** is that the challenger does not call $x = \mathsf{Inv}_{\mathsf{clr}}(\mathsf{pp}_{\mathsf{clr}}, t, c_1)$ for decryption queries anymore, but $x = \mathsf{Inv}_{\mathsf{abo}}(\mathsf{pp}_{\mathsf{abo}}, t^*, \mathsf{vk}_\sigma, c_2)$. Namely, it has stored the ABO trapdoor t^* during the INIT phase and uses this to decrypt the ciphertext. The other computations of the Decrypt algorithm remain the same. We remind that, as in the previous games, when it gets a decryption query with $\mathsf{vk}_\sigma = \mathsf{vk}^*_\sigma$ it outputs $\perp$ immediately.

Game-5: The final game differs from the previous one in the CHALLENGE phase. During the encryption of the challenge ciphertext the challenger does not call $b \overset{R}{\leftarrow} \mathcal{B}_\lambda$ but sets $c_3 = b = b^*$. Therefore, it computes the c_1 component using the lossy branch of the CLR-LTDF as $c_1 = \mathsf{Eval}_{\mathsf{clr}}(\mathsf{pp}_{\mathsf{clr}}, b^*, x)$.

We will prove the following lemmas that conclude the proof of theorem 13.

Lemma 14. *The advantage of any PPT attacker playing the* ***CL-CCA*** *game is the same as the advantage of the same attacker playing* ***Game-1****.*

Proof. This lemma is obvious, since the only difference in the two games is the time of when the $\mathsf{Setup}_\sigma(1^\lambda) \to (\mathsf{vk}^*_\sigma, \mathsf{sk}^*_\sigma)$ is called. In the first game it is called during the CHALLENGE phase while in the second during the INIT phase. The distribution of all inputs to the attacker is exactly the same. □

Lemma 15. *The advantage of any PPT attacker playing* ***Game-1*** *is negligibly close to the advantage of the same attacker playing* ***Game-2****, given that the signature scheme* $(\mathsf{Setup}_\sigma, \mathsf{Sign}_\sigma, \mathsf{Vrfy}_\sigma)$ *is OT-SUF secure.*

Proof. To prove this lemma, let E define the event that a PPT attacker $\mathcal{A}$ makes at least one decryption query $\mathsf{ct}_i = (c_0, c_1, c_2, c_3, \mathsf{vk}_\sigma, \sigma)$ such that $\mathsf{vk}_\sigma = \mathsf{vk}^*_\sigma$, the signature σ is valid, and the ciphertext $\mathsf{ct}_i \neq \mathsf{ct}^*$ (in case ct_i is done in PHASE-2). The two games differ only when this event happens. We claim that the probability of E is negligible in λ in both games, under the assumption that the signature scheme $(\mathsf{Setup}_\sigma, \mathsf{Sign}_\sigma, \mathsf{Vrfy}_\sigma)$ is OT-SUF secure. Therefore, the difference in the advantage is also negligible.

To prove this let's assume that the probability of E in one of the games is non-negligible. Then we can build a successful PPT attacker $\mathcal{B}$ on the signature scheme. $\mathcal{B}$ will simulate the security game and use $\mathcal{A}$ to break the scheme. First $\mathcal{B}$ receives a verification key vk' from its challenger. This key will take the place of vk^* in the security game. Then $\mathcal{B}$ proceeds in playing the security game until E happens (before that both games are exactly the same). That is, it calls $\mathsf{Setup}(1^\lambda)$, answers leakage queries, and answers decryption queries that do not satisfy the restrictions of the event E. $\mathcal{B}$ can do this since it has the secret key t. During the CHALLENGE phase, it creates a valid ciphertext ct^* with the verification key $\mathsf{vk}^* = \mathsf{vk}'$, but in order to create the signature σ^*, it queries its signature challenger with the message (c_0, c_1, c_2, c_3).

In case the event E happens in PHASE-1, the attacker $\mathcal{A}$ requested a decryption on the ciphertext $\mathsf{ct}_i = \big(c_0, c_1, c_2, c_3, \mathsf{vk}'_\sigma, \sigma\big)$ with $\mathsf{Vrfy}_\sigma(\mathsf{vk}', (c_0, c_1, c_2, c_3), \sigma) = \mathsf{true}$. At this point the simulator $\mathcal{B}$ outputs $((c_0, c_1, c_2, c_3), \sigma)$ as a valid forgery. In that case $\mathcal{B}$ breaks the OT-SUF scheme without having done any signature queries.

In case the event E happens in PHASE-2, we know that $\mathsf{ct}_i \neq \mathsf{ct}^*$. Again $\mathcal{B}$ outputs $((c_0, c_1, c_2, c_3), \sigma)$ as a forgery. Since this is different from the challenge ciphertext, it is also a successful forgery for the OT-SUF scheme. $\square$

Lemma 16. *The advantage of any PPT attacker playing **Game-2** is negligibly close to the advantage of the same attacker playing **Game-3**, given the hidden lossy branch property of the ABO-TDF* $(\mathsf{Setup}_{\mathsf{abo}}, \mathsf{Eval}_{\mathsf{abo}}, \mathsf{Inv}_{\mathsf{abo}})$.

Proof. We will prove this lemma by describing a simulator $\mathcal{B}$ that can distinguish the lossy branch from the injective one when given access to an oracle $\mathcal{O}$. The oracle takes as inputs two branches $b_0, b_1 \in \mathcal{B}'$ and outputs the public parameters of the ABO-TDF using one of the two branches with probability 1/2.

The simulator works by calling $\mathsf{Setup}_\sigma(1^\lambda) \to (\mathsf{vk}^*_\sigma, \mathsf{sk}^*_\sigma)$ and then calling the oracle with $\mathcal{O}(\mathsf{vk}^*_\sigma, 0^w) \to \mathsf{pp}_{\mathsf{abo}}$. All operations for the games are computed as defined except the $\mathsf{Setup}_{\mathsf{abo}}(1^\lambda, \cdot) \to (\mathsf{pp}_{\mathsf{abo}}, t^*)$. The simulator uses for $\mathsf{pp}_{\mathsf{abo}}$ the public parameters it received from the oracle. Notice that it knows the secret key t (but not t^*), and therefore it can answer all leakage and decryption queries. As before decryption queries with $\mathsf{vk}_\sigma = \mathsf{vk}^*_\sigma$ are answered with $\perp$.

Notice that, depending on the choice of the oracle, $\mathcal{B}$ simulates perfectly **Game-2** or **Game-3**. $\square$

Lemma 17. *The advantage of any PPT attacker playing **Game-3** is negligibly close to the advantage of the same attacker playing **Game-4** given that it is hard to sample a non-injective branch from the CLR-LTDF* $(\mathsf{Setup}_{\mathsf{clr}}, \mathsf{Eval}_{\mathsf{clr}}, \mathsf{Inv}_{\mathsf{clr}}, \mathsf{Update}_{\mathsf{clr}})$.

Proof. The two games differ solely on the computations during the decryption queries. We will prove that with 1 minus probability the outputs of the decryption queries are exactly the same. According to the property of the CLR-LTDF family the probability that one of the branches $c_3 = b$ in the decryption queries gives a non injective function is negligible in λ. Therefore we assume that all the decryption queries contain injective branches for the CLR-LTDF.

In both games when the decryption query has $\mathsf{vk}_\sigma = \mathsf{vk}^*_\sigma$ the output is the same: $\perp$. If this is not the case, the decryption algorithm calculates an x, either by $x = \mathsf{Inv}_{\mathsf{clr}}(\mathsf{pp}_{\mathsf{clr}}, t, c_1)$ in **Game-3** or $x = \mathsf{Inv}_{\mathsf{abo}}(\mathsf{pp}_{\mathsf{abo}}, t^*, \mathsf{vk}_\sigma, c_2)$ in **Game-4**. Then it asserts that $c_1 = \mathsf{Eval}_{\mathsf{clr}}(\mathsf{pp}_{\mathsf{clr}}, c_3, x)$ and $c_2 = \mathsf{Eval}_{\mathsf{abo}}(\mathsf{pp}_{\mathsf{abo}}, \mathsf{vk}_\sigma, x)$. According to the above assumption and since $\mathsf{vk}_\sigma \neq \mathsf{vk}^*_\sigma$, we know that both functions are injective. This means that for a pair (c_1, c_2) there is either a unique x that satisfies both relations or none. In **Game-3** this x is found by inverting the CLR-LTDF, while in **Game-4** it is found by inverting the ABO-TDF. If there is no such x one of the two checks will fail and in both games the challenger will output $\perp$. $\square$

Lemma 18. *The advantage of any PPT attacker playing **Game-4** is negligibly close to the advantage of the same attacker playing **Game-5**, given the indistinguishability of lossy branch under continual memory leakage of the CLR-LTDF* $(\mathsf{Setup}_{\mathsf{clr}}, \mathsf{Eval}_{\mathsf{clr}}, \mathsf{Inv}_{\mathsf{clr}}, \mathsf{Update}_{\mathsf{clr}})$.

Proof. In order to prove this lemma we will construct a PPT attacker $\mathcal{B}$ on the indistinguishability game of the CLR-LTDF if the difference in the advantage of any PPT attacker $\mathcal{A}$ is more than negligible. According to the lossy branch indistinguishability game the simulator/attacker $\mathcal{B}$ receives the public parameters $\mathsf{pp}_{\mathsf{clr}}$ from its challenger where $(\mathsf{pp}_{\mathsf{clr}}, t) \leftarrow \mathsf{Setup}_{\mathsf{clr}}(1^\lambda, b^*)$. Notice that the simulator does not know the lossy branch b^* or the trapdoor t.

However it simulates **Game-4** and **Game-5** by calling $(vk^*_\sigma, \mathsf{sk}^*\sigma) \leftarrow \mathsf{Setup}_\sigma(1^\lambda)$ and $(\mathsf{pp}_{\mathsf{abo}}, t^*) \leftarrow \mathsf{Setup}_{\mathsf{abo}}(1^\lambda, \mathsf{vk}^*_\sigma)$. Notice that it knows the trapdoor t^* and the verification key vk^*. Therefore it can answer the decryption queries according to the rules of the games. For the leakage queries on the trapdoor t, $\mathcal{B}$ forwards them to its challenger.

In the Challenge phase $\mathcal{B}$ requests the challenge branch from its challenger. It uses this in order to create the challenge ciphertext ct^*. If this branch is the injective branch b, $\mathcal{B}$ simulates **Game-4**. If it received the lossy branch b^*, it simulated **Game-5**. $\mathcal{B}$ can answer the remaining decryption queries as before. Notice that no more leakage queries are needed. □

Lemma 19. *No PPT attacker has a more than negligible advantage in **Game-5** given that $\mathcal{H}$ is a pairwise independent hash function family.*

Proof. In order to prove this lemma we will prove that the challenge ciphertext ct^*, encrypting either of the two messages, is within negligible statistical distance from a ciphertext, encrypting a random message. Therefore no PPT attacker can distinguish encryption of one message from the other with more than negligible advantage.

To do that fix all the randomness of **Game-5**, including the randomness of the adversary, except the randomness of choosing the variable x and the function h. We will show that conditioning on the values of the challenge ciphertext c_1^*, c_2^* and the value of the fixed randomness, the value $h(x)$ is a nearly uniform and independent "one-time pad". By averaging over the choice of the fixed randomness, we conclude that any adversary can not distinguish the encryption of the message m_c (for $c \in \{0,1\}$) from the encryption of a random message.

In order to do that we first observe that both Eval functions used in the challenge ciphertext and which contain x are lossy. Namely, $\mathsf{Eval}_{\mathsf{clr}}(\mathsf{pp}_{\mathsf{clr}}, b^*, \cdot)$ can take at most 2^{n-k} values, while $\mathsf{Eval}_{\mathsf{abo}}(\mathsf{pp}_{\mathsf{abo}}, \mathsf{vk}_\sigma^*, \cdot)$ can take at most $2^{n-k'}$ values. As a result the random variable (c_1^*, c_2^*) can take at most $2^{2n-k-k'} \leq 2^{n-\kappa}$ values according to the relation $k + k' \geq n + \kappa$.

As a result, we have that $\tilde{H}_\infty\left(x|(c_1^*, c_2^*, h)\right) \geq H(x|h) - (n - \kappa) = n - n + \kappa = \kappa$. We have that $H\left(x|h\right) = n$ since x and h are independent. Since $\ell \leq \kappa - 2\log(1/\epsilon)$ according to Lemma 1, we have that $(c_1^*, c_2^*, x, h(x))$ is within statistical distance $\epsilon = \mathsf{negl}(\lambda)$ from $(c_1^*, c_2^*, x, U_\ell)$. □