Witness Maps and Applications

Suvradip Chakraborty*, Manoj Prabhakaran**, and Daniel Wichs***

Abstract. We introduce the notion of *Witness Maps* as a cryptographic notion of a proof system. A *Unique Witness Map* (UWM) deterministically maps all witnesses for an **NP** statement to a single representative witness, resulting in a computationally sound, deterministic-prover, non-interactive witness independent proof system. A relaxation of UWM, called Compact Witness Map (CWM), maps all the witnesses to a small number of witnesses, resulting in a "lossy" deterministic-prover, non-interactive proof-system. We also define a *Dual Mode Witness Map* (DMWM) which adds an "extractable" mode to a CWM.

Our main construction is a DMWM for all **NP** relations, assuming sub-exponentially secure indistinguishability obfuscation ($i\mathcal{O}$), along with standard cryptographic assumptions. The DMWM construction relies on a CWM and a new primitive called *Cumulative All-Lossy-But-One Trapdoor Functions* (C-ALBO-TDF), both of which are in turn instantiated based on $i\mathcal{O}$ and other primitives. Our instantiation of a CWM is in fact a UWM; in turn, we show that a UWM implies Witness Encryption. Along the way to constructing UWM and C-ALBO-TDF, we also construct, from standard assumptions, *Puncturable Digital Signatures* and a new primitive called *Cumulative Lossy Trapdoor Functions* (C-LTDF). The former improves up on a construction of Bellare et al. (Eurocrypt 2016), who relied on sub-exponentially secure $i\mathcal{O}$ and sub-exponentially secure OWF.

As an application of our constructions, we show how to use a DMWM to construct the first *leakage and tamper-resilient signatures* with a *deterministic signer*, thereby solving a decade old open problem posed by Katz and Vaikunthanathan (Asiacrypt 2009), by Boyle, Segev and Wichs (Eurocrypt 2011), as well as by Faonio and Venturi (Asiacrypt 2016). Our construction achieves the optimal leakage rate of $1 - o(1)$.

Keywords: Witness Maps, Zero Knowledge, Lossy Trapdoor Functions, Leakage, Tampering, Signatures.

* Institute of Science and Technology Austria. Work carried out while at IIT Madras. suvradip.chakraborty@ist.ac.at

** Indian Institute of Technology Bombay, India. Supported by the Dept. of Science and Technology, India via the Ramanujan Fellowship and an Indo-Israel Joint Research Project grant, 2018. mp@cse.iitb.ac.in

*** Northeastern and NTT Research. Research supported by NSF grants CNS-1314722, CNS-1413964, CNS-1750795 and the Alfred P. Sloan Research Fellowship. wichs@ccs.neu.edu

1 Introduction

A foundational innovation of theoretical computer science has been the generalization of the notion of what a *proof* is. Interactive proofs, zero-knowledge proofs and probabilistically checkable proofs are all critical to the current theory – and practice – of computer science. In this work, we introduce and explore yet another notion of a proof, against the backdrop of recent advances in cryptography.

A conventional proof of a statement that can be verified by an efficient program is called a *witness* for the statement. Goldwasser, Micali and Rackoff, in their seminal work on interactive proofs [30], introduced the fascinating concept of zero-knowledge proof protocols which reveal no "knowledge" about the witness to a verifier, yet can soundly convince her of the existence of a witness. The notion of knowledge was formalized using *simulators*. An important direction of subsequent investigation has been to develop more rudimentary models of proofs, which when realized, offer powerful cryptographic applications. In particular, Blum, Feldman and Micali [6] introduced the notion of *non-interactive* zero-knowledge proofs (NIZK), wherein they reverted to the conventional notion of a proof being just a single message that the prover can send to the verifier, but allowed a "trusted setup" in the form of a common reference string, with respect to which the proof would be verified. Feige and Shamir [23] defined *witness indistinguishability* as a simpler notion of hiding information about the witness.

> *The central object we investigate in this paper – called a Witness Map – is an even more rudimentary notion of a proof, wherein a proof is simply an alternate representation of a witness, verified using an alternate relation.*

The prover and the verifier are required to be efficient and *deterministic*, and the proof system is required to be computationally sound. A common reference string is used to generate and verify the proofs. Instead of zero-knowledge property, we require a "lossiness" property. Specifically, in a *Compact Witness Map* (CWM), each statement has a small number of proofs that its witnesses could map to, with an important special case being that of a *Unique Witness Map* (UWM).

One may wonder if it is possible to hide the witness to any extent at all, when the prover is deterministic. But we show that if indistinguishability obfuscation ($i\mathcal{O}$) and one-way functions exist, then UWMs do exist. On the other hand, we show that the existence of UWMs imply the existence of Witness Encryption (WE). Hence UWM could be viewed as the newest member of "obfustopia," and arguably the one with the simplest definition.[1]

We extend the scope of witness maps further to define the notion of a *Dual Mode Witness Map* (DMWM). In a DMWM, a proof either allows the original witness to

[1] We present a brief formulation (omitting some formalism) here. A UWM for an **NP** language L is specified by a distribution over polynomial time verifiable relations R^{K}, such that (1) for every $x \in L$, there is a canonical witness $w^*_{\mathsf{K},x}$ with $(x, w^*_{\mathsf{K},x}) \in R^{\mathsf{K}}$, which can be efficiently computed from any witness w for $x \in L$, and (2) it is computationally infeasible to find a pair $(x, w^*) \in R^{\mathsf{K}}$ such that $x \notin L$.

be extracted (using a trapdoor) or it is lossy. Which mode a proof falls into depends on whether or not the "tag" used for constructing the proof equals a hidden tag used to derive the mapping key. In defining the lossy mode, we introduce a strong form of lossiness – called *cumulative lossiness* – which bounds the total amount of information about a witness that can be revealed by *all* the proofs using all the lossy tags. We also show how to construct a DMWM for any NP relation using a CWM and a new notion of lossy trapdoor functions (which may be of independent interest).

We show that DMWMs can be readily used to solve an open problem in the area of leakage-resilient cryptography, namely, that of constructing a leakage and tamper resilient signature scheme (where all the data and randomness used by the signer are open to leakage and tampering). A crucial aspect of our construction that helps in achieving this is that signing algorithm in our scheme is deterministic, a property it inherits from the prover in a DMWM. We also extend our results to a *continuous* leakage and tampering model.

We expand on each of these contributions in greater detail below.

1.1 Witness Maps

We introduce a new primitive called a compact/unique witness map (CWM/UWM). Informally, CWM/UWM deterministically maps all possible valid witnesses for some **NP** statement to a much smaller number of *representative witnesses*, resulting in loss of information regarding the original witness. Nevertheless, the mapping should preserve the functionality of the witnesses, namely that the representative witnesses should be efficiently verifiable and (computationally) guarantee the soundness of the statement. A particularly strong form of CWM is a Unique Witness Map (UWM), in which all the possible witnesses for a statement are mapped to a single representative witness. In other words, in a UWM the representative witness only depends on the statement being proved, but not which of the original witnesses was used to prove it.[2] While we require the CWM/UWM to be deterministic, it can depend on some public *common reference string* (CRS). A UWM is essentially equivalent to a non-interactive witness indistinguishable argument (in the CRS model) with a deterministic prover and a deterministic verifier.

Defining CWM/UWM. In more detail, a CWM consists of three algorithms (setup, map, check). The setup algorithm generates a CRS K. The deterministic algorithm $\mathsf{map}(\mathsf{K}, x, w)$ takes as input a statement x and a witness w and maps it to a representative witness w^*. The algorithm $\mathsf{check}(\mathsf{K}, x, w^*)$ takes as input the statement x and the representative witness w^* and outputs 1 if it verifies and 0 otherwise. We require the standard completeness property (if w is good witness for x then $\mathsf{check}(\mathsf{K}, \mathsf{map}(\mathsf{K}, x, w)) = 1$) and computational soundness (if x is false then it's computationally hard to produce w^* such that $\mathsf{check}(\mathsf{K}, x, w^*) = 1$). Lastly, we require that for any true statement x the set of

[2] Note that uniqueness is a property of the map/prover, but we do not require uniqueness for the verifier; for any given statement, there may be many representative witnesses that the verifier would accept, but the map/prover always produces a unique one.

possible representative witnesses $\{w^* = \mathsf{map}(\mathsf{K}, x, w) \ : \ w \text{ witness for } x\}$ is small, and potentially much smaller than the set of all original witnesses w for x. In a UWM, the set of representative witnesses needs to be of size 1, meaning there is a unique representative witness for each x in the language.

Constructing UWM. We give a simple construction of a UWM from $i\mathcal{O}$ and a punctured digital signature (PDS) scheme (see below), by leveraging the framework of Sahai and Waters [49] previously used to construct NIZKs. Our construction could be seen as implementing "deterministic witness signatures," wherein the signing key is a valid witness to a statement. We remark that a notion of witness signatures exists in the literature [32], building on the notion of "Signatures of Knowledge" [13]; however, these are incomparable to our UWM construction, as they allow randomized provers, but demand extractability of the witness (and in the case of Signatures of Knowledge, simulatability as well).

Puncturable Digital Signatures (PDS). As part of our UWM construction, we rely on Puncturable Digital Signatures (PDS). This primitive allows us to create a punctured signing key that cannot be used to sign some specified message m but otherwise correctly produces signatures for all other messages $m' \neq m$. We improve upon the construction of PDS by Bellare et al. [5], who relied on *sub-exponentially secure* Indistinguishability Obfuscation and*sub-exponentially secure* one-way functions (OWF). Our construction shows that PDS is equivalent to OWF.

Implications of UWM. We show that UWMs are a powerful primitive and, in particular, imply witness encryption (WE) [25]. However, we do not know of any such implication for CWMs in general, especially if the image size of the map can be (slightly) super-polynomial.

Dual-Mode Witness Maps. We also introduce a generalization of compact/unique witness maps (CWM/UWM) that we call *dual-mode witness maps* (DMWM). In a DMWM the map and check algorithms take as input an additional tag or branch parameter b. Furthermore the setup algorithm also takes as input a special "injective branch" b^* which is used to generate the CRS along with a trapdoor td. If $b = b^*$ then the map is injective and the original witness w can be extracted from the representative witness w^* output by the map using the trapdoor td. On the other hand, the maps for all $b \neq b^*$ is *cumulatively lossy* – i.e., even taken together, they do not reveal much information about the original witness. The identity of the injective branch b^* is hidden by the CRS.

Our definition of the cumulative lossiness property for DMWM is motivated by its application to leakage and tamper resilient signatures (see below). But it is in itself a property that can be applied more broadly. In particular, we introduce the following primitives and employ them in our construction of DMWMs (in combination with CWMs).

Cumulatively Lossy Trapdoor Functions. We introduce new variants of *lossy trapdoor functions* (LTDFs) [45], which we call *cumulatively lossy trapdoor functions* (C-LTDFs). Recall that, in an LTDF, a function f can be sampled to either be injective

(and the sampling algorithm also generates an inversion trapdoor) or lossy (the image of f is substantially smaller than the input domain) and the two modes should be indistinguishable. For C-LTDFs, we further require that arbitrarily many lossy functions taken together are jointly lossy. In other words, if we sample arbitrarily many independent lossy functions f_i then their concatenation $(f_1, \ldots, f_\ell)(x) = (f_1(x), \ldots, f_\ell(x))$ is also lossy. We can construct C-LTDFs from DDH or LWE.

We also define *cumulatively all-lossy-but-one trapdoor functions* (C-ALBO-TDFs). This is a collection of functions $f(b, \cdot)$ parametrized by a *branch index* b. We can sample f with a special injective branch b^* such that $f(b^*, \cdot)$ is injective (and we have the corresponding inversion trapdoor) but $f(b, \cdot)$ is lossy for all $b \neq b^*$. We should not be able to distinguish which branch is the injective one. Furthermore, the lossy branches $b \neq b^*$ are cumulatively lossy. Previous constructions of LTDFs with branches [45] only achieved the opposite notion of "all-but-one lossy", where there is one lossy branch and all the other branches are injective. To the best of our knowledge, constructing ALBO-LTDFs (even without the cumulative loss requirement) was previously open. We show how to boost C-LTDFs to get C-ALBO-LTDFs via $i\mathcal{O}$.

1.2 Application: Leakage and Tamper Resilient Signatures

A digital signature scheme is one of the most fundamental cryptographic primitives and is used as an important building block in many cryptographic protocols and applications. Signature schemes are used ubiquitously in practice, in a variety of settings and applications. In particular, signing keys are often embedded in smart cards and devices operated by untrusted users. Such settings admit powerful "physical attacks" exploiting numerous side-channels for leaking (e.g. power analysis, timing measurements, microwave attacks [37,38]) and tampering (see for instance [8,47]). This has led to several works over the last decade that addressed security of cryptographic primitives – and in particular of digital signature schemes – that are leakage and/or tamper resistant [11,15,21,35,39]. In this work, we address an important question that this body of work has raised again and again:

Is there a leakage and tamper resilient (LTR) signature scheme? Is there one with a *deterministic* signing algorithm?

The significance of this question lies in the fact that it appears harder to protect against an adversary who can target the randomness used in the scheme. When the randomness is open to attacks, current state of the art can protect only against leakage attacks [11, 14, 19], and not against tampering attacks (as explicitly posed in [21]). Note that if the adversary can obtain signatures produced using arbitrarily tampered randomness, it can set the randomness to a constant (say, all 0s) and therefore effectively make the signing algorithm deterministic. Therefore, a natural solution is to *entirely eliminate attacks on the randomness* by constructing a LTR signature scheme with a deterministic signing algorithm. Indeed, this is the approach taken in [14], but unfortunately their solution does not offer security against tampering of the secret key.

LTR Signature Results. Our main contribution is the construction of a *leakage and tamper resilient (LTR) signature scheme* with a deterministic signing algorithm. We focus on the *bounded* leakage and tampering model of Damgård et al. [15]. In this model, the adversary can get some bounded amount of leakage on the secret key and can also tamper with the secret key some bounded number of times; these bounds can be made arbitrarily large but have to be chosen a-priori. We strengthen the model so that *only publicly known, fixed components of the scheme (namely, the code and public parameters) are fully protected.* In particular, any randomness used during computation is subject to leakage *and* tampering. The key-generation phase is also subject to leakage (but is protected from tampering). Note that tamperability of the signing randomness invalidates prior results [15, 21], and motivates the need for finding a deterministic solution. A recent work of Chen et al. [14] constructs a deterministic leakage-resilient (but *not* tamper-resilient) signature scheme from $i\mathcal{O}$ and puncturable primitives. However, as we argue later, this construction does not generalize to the setting of tampering.

Our schemes achieve a leakage rate of $1-o(1)$, where the leakage rate is defined as the ratio of the amount of leakage to the size of the secret signing key. The scheme natively only achieves selective security, where the message to be forged is chosen by the adversary at the very beginning of the attack game. Adaptive security follows via complexity leveraging. We present our construction using generic primitives discussed below. While current instantiations of these primitives rely on indistinguishability obfuscation ($i\mathcal{O}$) and either DDH or LWE, there is hope that our template can also be instantiated under weaker assumptions in the future. Our construction combines ideas from leakage-resilience [11] and tamper-resilience [21], but replaces various ingredients with our new building blocks to facilitate a deterministic solution.

We also discuss how to extend our results to the *continuous* leakage and tampering model. In this mode, the key is periodically refreshed and the adversary is only bounded in the amount of leakage and tampering that can be performed in each time period, but can continuously attack the system for arbitrarily many time periods. However, in this model, we inherently cannot allow tampering of the randomness used to perform the refreshes.

Along the way toward our main result for LTR Signatures, we introduce several new cryptographic primitives and constructions, which may be of independent interest and which we now proceed to describe.

Construction Outline. We construct deterministic leakage and tamper resilient signatures directly from dual-mode witness maps (DMWM) and a leakage-resilient one-way function (which, as we shall see, can be based on general one-way functions). As mentioned above, we construct DMWMs by combining a compact witness map (CWM) for **NP**, with a C-ALBO-LTDF, constructing other primitives like PDS and C-LTDF along the way. While current instantiations of CWMs and C-ALBO-LTDFs rely on strong assumptions (i.e., $i\mathcal{O}$ and either DDH or LWE), this does not appear inherent and there is hope that future work can find alternate instantiations based on weaker assumptions.

In particular, while UWMs imply a strong primitive (namely, Witness Encryption), the same is not known for DMWM, CWM or C-ALBO-LTDF.

1.2.1 Related Work on Leakage & Tamper-Resilient Signatures Various notions of leakage-resilient signatures (LRS) have been studied for about a decade now. Alwen, Dodis and Wichs [2] and Katz and Vaikuntanathan [35] gave initial constructions of LRS schemes in the bounded leakage model, where the leakage is allowed to happen from the entire memory of the device. The construction of [2] was in the random oracle (RO) model. [35] gave a standard model construction, which had a deterministic signing scheme as well, but which allowed only a logarithmic number of signature queries, and the total leakage allowed degraded with number of queries. Meanwhile, Faust, Kiltz, Pietrzak and Rothblum [22] gave a construction of a *stateful* LRS scheme in the "Only Computation Leaks" model of Micali and Reyzin [41]. The first full-fledged construction of fully leakage-resilient (FLR) signatures – which allowed bounded leakage from the randomness used for key-generation and signing – were proposed independently by Boyle et al. [11] and Malkin et al. [39]. Faonio et al. [19] also gave a construction of FLR signatures in the bounded retrieval model, where the secret key (and the leakage from it) may be larger than the size of a signature. In this setting, standard existential unforgeability is impossible to achieve, since the adversary can simply leak a forgery. Hence the authors only demand a graceful degradation of security to hold. Yuen et al. [50] constructed a FLR signature scheme in the selective auxiliary input leakage model, where it is assumed that the leakage is a computationally hard-to-invert function. The recent work of Chen et al. [14] gave an FLR signature scheme with a deterministic signing algorithm, and achieved selective unforgeability, relying on $i\mathcal{O}$.

Tamper resilience was addressed in [15,21]. The question of *fully* leakage and tamper resilient signatures (i.e., allowing leakage from and tampering of randomness as well as secret key) was explicitly posed as an open problem in [21]. The continual memory leakage (CML) model has been studied in [12,16,39].

Comparison with the work of [14]. Recently, Chen et al. [14] constructed a deterministic leakage-resilient (but *not* tamper-resilient) signature scheme in the bounded leakage model. An important limitation of their construction is that it does not appear amenable to a *leakage-to-tamper reduction*, which relies on being able to bound the amount of information revealed by a signature using the tampered signing key, given the verification key. (Their signing key sk is a ciphertext of a symmetric-key encryption scheme and the verification key vk comprises of two obfuscated programs.)

Comparison with the work of [20]. Predictable argument of knowledge (PAoK) [20] are 2-round public-coin argument systems where the answer of the prover can be predicted, given the private randomness of the verifier (thus necessitating the prover to be *deterministic*). They insist on knowledge soundness from PAoK and show that a PAoK for general **NP** relations is equivalent to extractable witness encryption. In contrast, DMWM are *non-interactive*.

1.3 Technical Overview

1.3.1 Compact Witness Maps. We now sketch the main idea behind the construction of our unique witness map (UWM) scheme, which is the strongest form of compact witness maps (CWMs). Our construction essentially follows the same (abstracted out) approach of Sahai and Waters NIZKs [49]. The setup of the UWM generates a (public) CRS K. The CRS K in our construction embeds the description of an obfuscated program P, with the signing key of the Puncturable Digital Signature (PDS) scheme hard-coded in it. The obfuscated program P functions as follows: the input to the program P is a statement-witness pair, say (stmnt, w) belonging to underlying **NP** relation R_ℓ (we consider statements of size at most ℓ). The program simply checks if $R_\ell(\mathsf{stmnt}, w) = 1$, and signs the statement stmnt using the signing key sk to obtain a signature on stmnt. While generating the mapping, the mapping algorithm $\text{UWM}.\mathsf{map}(\mathsf{K}, \mathsf{stmnt}, w)$ runs the obfuscated program P with input (stmnt, w) to obtain a signature σ_{stmnt} on stmnt using sk. The representative witness w^* is just the signature σ_{stmnt}. The verification of the mapping is done by simply verifying the signature σ_{stmnt} (using the verification algorithm of the PDS scheme).

For proving security of the UWM scheme, we consider the notion of selective soundness[3], where the adversary announces the statement stmnt^* on which it tries to break the soundness (i.e, produce a representative witness w^*corresponding to it) of the UWM scheme, before receiving the key K. In the hybrid, we change the obfuscated program by puncturing the signing key sk at the statement stmnt^*. The consistency property of the PDS scheme ensures that the signatures output by the punctured key sk_{stmnt^*} (punctured at stmnt^*) produces the same output as the signatures generated by the original signing key sk. If the adversary could produce a witness w^* (which is nothing but a signature) corresponding to the false statement stmnt^*, this means it has managed to successfully output a forgery for the PDS scheme. Also note that, our mapping satisfies uniqueness, since (x, w) is deterministically mapped to the signature on x, independent of w.

Construction of PDS. To instantiate the UWMs described above, it remains to construct a Puncturable Digital Signature (PDS) scheme. The work of Sahai and Waters [49] implicitly constructs one using iO as a part of their construction of NIZKs, and Bellare et al. [5] makes this explicit. We show a simple construction from one-way functions. The main idea is to rely on *tree-based signatures*, where every node of the tree is associated with a fresh verification/signing key of a standard (one-time) signature and a PRG seed; the seed of the parent node is used to generate the values (the verification/signing key and the seed) of each of the two children nodes. The verification key of the scheme corresponds to that of the root note and the signing key corresponds to the (signing key, seed) of the root. Each message traces out a path in the tree from a root to a leaf and the signature corresponds to a "certificate chain" consisting of signed verification keys

[3] The size of the statements supported by UWM scheme is bounded (looking ahead, this will indeed be the case in our FLTR signature scheme). Hence, we can achieve adaptive soundness via a standard complexity leveraging argument, albeit incurring a sub-exponential loss in the security parameter.

along that path together with a signature of the message under the leaf's key. Note that the intermediate values in the tree are generated on the fly and the entire tree (which is of exponential size) is never stored all at once. Puncturing the signing key is analogous to puncturing the GGM PRF [9, 10, 27, 36]. In particular, we remove all of the values along one path from the root to a particular leaf for the specified message on which we are puncturing, and instead give out the values of (signing key, seed) for each sibling along that path; this is sufficient to generate signatures for every other message aside from the punctured one.

UWMs imply Witness Encryption. Lastly, we show that UWMs are a powerful cryptographic primitive and in fact imply *witness encryption* (WE) [25]. In a WE scheme, it is possible to encrypt a message m under an **NP** statement x such that, if the statement is true, then the ciphertext can be decrypted using any witness w for x. However, if x is a false statement, then the ciphertext should computationally hide the encrypted message. To construct a WE scheme from a UWM the encryption algorithm chooses a random seed z for a pseudorandom generator G and sets $y = G(z)$. It then uses a UWM to get a representative witness w^* for the statement $\hat{x}$ stating that "either x is true or y is pseudorandom", using z as the witness. It uses the Goldreich-Levin hardcore bit of w^* to blind the message m and outputs the blinded value along with y. The decryption algorithm uses the UWM to map the witness w for x into the unique witness w^* for the statement $\hat{x}$. It then computes the hardcore bit of w^* and uses it to recover the message. Intuitively, if an adversary can break WE security, then it can distinguish encryptions of 0 and 1 with non-negligible probability even if x is a false statement. This means that, using Goldreich-Levin decoding, it can compute the correct value w^* given y with non-negligible probability. Furthermore this value w^* is a valid representative witness for the statement $\hat{x}$. Since the adversary cannot break the PRG, it must also compute a valid representative witness for $\hat{x}$ if we switch y to false. But this contradicts the soundness of UWM.

1.3.2 Leakage and Tamper Resilient Signatures.

We now give an overview of our leakage and tamper resilient (LTR) signature construction. The construction proceeds in 3 steps. First, we construct LTR signatures from dual-mode witness maps (DMWMs). Second, we construct DWMs from cummulatively all-lossy-but-one tradoor functions (C-ALBO-TDFs) and compact witness maps (CWMs). Thirdly, we construct C-ALBO-TDFS from DDH/LWE and iO.

LTR Signatures from DMWMs. Recall that DMWM is essentially a witness map that takes as input a branch index b. The CRS is also generated with an injective branch b^* and a trapdoor td. If the map uses the branch $b = b^*$ then it is injective and the original witness can be extracted using the trapdoor. Otherwise the map reveals very little information about the original witness. The two modes are computationally indistinguishable from each other.

Our signature scheme has the following form: The signing key is a random string x, and the verification key is $y = H(x)$, where H is a sufficiently compressing, second pre-image resistant hash function. To sign a message m, we set the branch for the DMWM to

be the message m, and construct a representative witness w^* for the statement: $\exists x, y = H(x)$ using x as the original witness. Note that the signing procedure is deterministic. The verifier checks the representative witness using the DMWM scheme.

To argue selective security, we can set up the CRS of the DMWM so that the injective branch b^* is exactly the message that the adversary will forge the signature on. It remains indistinguishable to the adversary that this happened and hence the probability of forging does not change. However, now we can extract a pre-image x' such that $H(x') = y$ from the adversary's forgery. Moreover, since all the other signatures obtained by the adversary are all lossy, it would be *information-theoretically* hard to recover the original pre-image x. This holds even given some bounded additional leakage about the secret key x. It also holds even if x is tampered and then used to produce a signature since this still only provides bounded leakage on x. Therefore we recover a second pre-image $x' \neq x$ which contradicts the second pre-image resistance of H.

We also adapt our results to the continuous leakage and tampering (CLT) model. We do so by essentially taking the same construction, but using a "entropy-bounded" or "noisy" continuous-leakage-resilient (CLR) one-way relation [16] in place of the second pre-image resistant hash (which can be thought of as a leakage-resilient one-way function). We achieve security as long as the adversary cannot tamper the randomness of the refresh procedure, and this restriction is inherent.

DMWMs from CWMs via C-ALBO-TDFs. We now discuss how to construct dual-mode witness maps (DMWMs) from compact witness maps (CWMs). Recall that DMWM has branches in one of two modes: injective and lossy. On the other hand a CWM does not have any branches and is always lossy. To convert a CWM into DMWM we add a "cumulatively all-lossy-but-one trapdoor functions (C-ALBO-TDFs)". This is a family of functions $f(b, \cdot)$ parametrized by tags/branches b such that, for one special branch b^* the function $f(b^*, \cdot)$ is injective and efficiently invertible using a trapdoor, but for all other $b \neq b^*$ the functions $f(b, \cdot)$ are cumulatively lossy. The CRS of the DMWM will consist of the public key of the C-ALBO-TDF with the special injective branch b^* as well as a CRS of CWM scheme. To compute a proof for a statement y with witness w under a tag b, the prover computes $z = f(b, w)$ and then uses the CWM to prove that z was computed correctly using a valid witness w for the statement y.

Construction of C-ALBO-TDFs. Finally, we discuss how to construct *cumulative all-lossy-but-one* trapdoor functions (C-ALBO-LTDFs). We start with a simpler primitive of C-LTDFs which can be used to sample a function f_{ek} described by a public key ek. The key ek can be sampled indistinguishably in either lossy or injective mode (with a trapdoor). We require that the combination of arbitrarily many different lossy functions is cumulatively lossy.

We construct C-LTDFs by adapting a construction of LTDFs from DDH due to [45]. In that construction, the key ek is given by a matrix of group elements $g^{\boldsymbol{M}}$ where g is a generator of the group of order q and $\boldsymbol{M} \in \mathbb{Z}_q^{n \times n}$ is a matrix of exponents. For $\boldsymbol{x} \in \{0,1\}^n$ the function is defined as $f_{\mathsf{ek}}(\boldsymbol{x}) = g^{\boldsymbol{M} \cdot \boldsymbol{x}}$. If $\boldsymbol{M}$ is invertible than this function is injective and can be inverted with knowledge of $\boldsymbol{M}^{-1}$. If $\boldsymbol{M}$ is low rank (e.g., rank 1) then this

function is lossy. The two modes are indistinguishable by DDH. However, if we choose many different lossy functions by choosing random rank 1 matrices each time then the scheme is not cumulatively lossy; in fact n random lossy function taken together are injective! To get a cumulative lossy scheme, we fix some public parameters $g^{\boldsymbol{A}}$ where $\boldsymbol{A} \in \mathbb{Z}_q^{n \times n}$ is a random rank 1 matrix. We then choose each fresh lossy key ek by choosing a random $\boldsymbol{R} \in \mathbb{Z}_q^{n \times n}$ and setting $\mathsf{ek} = g^{\boldsymbol{RA}}$. Injective keys ek are still chosen as $g^{\boldsymbol{M}}$ for a random $\boldsymbol{M}$, which is invertible with overwhelming probability. It's easy to show that lossy and injective keys are indistinguishable even given the public parameters. Now if we apply many different lossy functions on the same input $\boldsymbol{x}$ we only reveal $\boldsymbol{Ax}$, which loses information about $\boldsymbol{x}$.

The above construction can also be extended to rely on the d-Linear assumption for larger d instead of DDH. We also provide an analogous construction under LWE by adapting an LTDF of [3], which relies on the "lossy mode" of LWE from [29].

We then show how to bootstrap C-LTDFs to get C-ALBO-LTDFS via iO. The idea is to obfuscate a program that, on input a branch b, applies a pseudorandom function to b to sample a fresh lossy key of a C-LTDF, except for a special branch b^* on which it outputs a (hard-coded) injective C-LTDF key. By relying on standard puncturing techniques, we show that this yields a C-ALBO-LTDF.

2 Preliminaries and Tools.

Some Notations For $n \in \mathbb{N}$, we write $[n] = \{1, 2, \cdots, n\}$. If x is a string , we denote $|x|$ as the length of x. For a distribution or random variable X, we denote $x \leftarrow X$ the action of sampling an element x according to X. When A is an algorithm, we write $y \leftarrow A(x)$ to denote a run of A on input x and output y; if A is randomized, then y is a random variable and $A(x; r)$ denotes a run of A on input x and randomness r. An algorithm A is probabilistic polynomial-time (PPT) if A is randomized and for any input $x, r \in \{0,1\}^*$; the computation of $A(x; r)$ terminates in at most $poly(|x|)$ steps. For a set S, we let U_S denote the uniform distribution over S. For an integer $\alpha \in \mathbb{N}$, let U_α denote the uniform distribution over $\{0,1\}^\alpha$, the bit strings of length α. Throughout this paper, we denote the security parameter by κ.

2.1 Basics of Information Theory

Here we give some basic definitions related to information theory needed for the formal proofs of some of our theorem.

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

A distribution supported on $\{0,1\}^n$ with min-entropy k is said to be an (n, k)-source.

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

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

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

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

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

Lemma 2. [18] *For any random variable X, Y and Z, if Y takes on values in $\{0,1\}^\ell$, then*

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

2.2 Required Primitives.

1. **(Puncturable Pseudo-Random Functions).** A puncturable pseudorandom function (pPRF) [9, 10, 36] is a PRF which lets one modify a given key by "puncturing" it at any input x, so that the punctured key lets one evaluate it on all inputs except at x. The output at x remains pseudorandom even given a punctured key.We follow the definition given in [33]. A *puncturable* PRF $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is given by a pair of polynomial time algorithms $(F.\mathsf{puncture}, F.\mathsf{eval})$ (as defined below) and equipped with an additional (punctured) key space $\widehat{\mathcal{K}}$.

   - $F.\mathsf{puncture}(K, T)$: This is a deterministic algorithm[4] that takes as input a key $K \in \mathcal{K}$, and a (polynomially bounded) set of inputs $T \subseteq \mathcal{X}$, and outputs a "punctured key" $\widehat{K}_T \in \widehat{\mathcal{K}}$ (punctured at the points in T).
   - $F.\mathsf{eval}(\widehat{K}_T, x)$: This is a deterministic algorithm that takes as input the punctured key $\widehat{K}_T \in \widehat{\mathcal{K}}$, and an input $x \in \mathcal{X}$. The correctness guarantee stipulates that for all $K \in \mathcal{K}$, $T \subseteq \mathcal{X}$, and $x \in \mathcal{X}$:

$$F.\mathsf{eval}(F.\mathsf{puncture}(K,T), x) = \begin{cases} F(K,x) & \text{if } x \notin T \\ \perp & \text{otherwise.} \end{cases}$$

[4] Requiring puncturing to be deterministic is w.l.o.g, because one can derandomize it by generating its random bits using a PRF that is keyed by a part of the key K on input T.

Security of Puncturable PRFs: The security of puncturable PRFs is captured by a game between a challenger and an adversary. The security game consists of the following four stages:

(a) **Setup Phase:** The challenger chooses uniformly at random a (master) PRF key $K \in \mathcal{K}$ and a bit $b \in \{0,1\}$.

(b) **Evaluation Query Phase:** In this phase the adversary $\mathcal{A}$ queries with a point $x \in \mathcal{X}$. The challenger sends back the evaluation $F(K,x)$ to $\mathcal{A}$. These queries can be made arbitrarily and adaptively by $\mathcal{A}$ polynomially many times.

(c) **Challenge Phase:** In this phase, the adversary $\mathcal{A}$ chooses a challenge point $T \subseteq \mathcal{X}$. $\mathcal{C}$ computes $\widehat{K}_T \leftarrow F.\mathsf{puncture}(K,T)$. For each $x \in T$, the challenger defines y_x as follows: if $b = 0$, $y_x = F(K,x)$ and else $y_x \leftarrow \mathcal{Y}$ is uniformly and independently sampled. $\mathcal{C}$ sends $(\widehat{K}_T, \{y_x\}_{x\in T})$ to $\mathcal{A}$.

(d) **Guess Phase:** $\mathcal{A}$ continues to query F, and outputs a guess b' for the bit b chosen by the challenger. Let $E \subseteq \mathcal{X}$ be the set of evaluation queries to F through out the game.

The output of the game is defined to be 1 iff $b = b'$ and $T \cap E = \emptyset$. The advantage of $\mathcal{A}$ in the above game, $\mathsf{Adv}^{\mathrm{pPRF}}_{F,\mathcal{A}}(\kappa)$ is defined as $\max\{0, p - \frac{1}{2}\}$, where p is the probability that the output of the game is 1 when $\mathcal{A}$ plays it.

Definition 3. *A pPRF scheme F is said to be secure if for all PPT adversaries $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{pPRF}}_{F,\mathcal{A}}(\kappa)$ is negligible in κ.*

2. **(Indistinguishability Obfuscation).** A uniform PPT machine $i\mathcal{O}$ is called an indistinguishability obfuscator for a circuit class $\{\mathcal{C}_\kappa\}_{\kappa\in\mathbb{N}}$ if it satisfies the following conditions:

(a) (*Functionality*). For all security parameters $\kappa \in \mathbb{N}$, for all $C \in \mathcal{C}_\kappa$, for all inputs x, we have that: $\Pr\left[C'(x) = C(x) : C' \leftarrow i\mathcal{O}(\kappa, C)\right] = 1$.

(b) (*Indistinguishability*). For any (not necessarily uniform) PPT adversaries $Samp$, $\mathcal{D}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that the following holds: if for all security parameters $\kappa \in \mathbb{N}$, $\Pr\left[\forall x, C_0(x) = C_1(x) : (C_0, C_1, \mathsf{st}) \leftarrow Samp(\kappa)\right] > 1 - \mathsf{negl}(\kappa)$, then the advantage of $\mathcal{D}$, denoted by $\mathsf{Adv}^{i\mathcal{O}}_{Samp,\mathcal{D}}(\kappa) = \left|\Pr\left[\mathcal{D}(\mathsf{st}, i\mathcal{O}(\kappa, C_b)) = b : (C_0, C_1, \mathsf{st}) \leftarrow Samp(\kappa), b \leftarrow \{0,1\}\right] - \frac{1}{2}\right|$ is negligible.

Let $\delta : \mathbf{N} \to \mathbf{R}$. We say that $i\mathcal{O}$ is δ-secure if for every PPT adversaries $Samp$, $\mathcal{D}$, it holds that $\mathsf{Adv}^{i\mathcal{O}}_{Samp,\mathcal{D}}(\kappa) \leq \delta(\kappa)$. We say that $i\mathcal{O}$ is *sub-exponentially secure* if it is $2^{-\kappa^\epsilon}$-secure, for some $0 < \epsilon < 1$.

3. **(Second Pre-image Resistance).** A family $\mathrm{SPR} = (\mathrm{SPR.gen}, \mathrm{SPR.eval})$ of efficiently computable functions from $\{0,1\}^{n(\kappa)}$ to $\{0,1\}^{m(\kappa)}$ is *second-preimage resistant* if for any PPT adversary $\mathcal{A}$ it holds that:

$$\Pr\left[\text{SPR.}\mathsf{eval}_s(x') = \text{SPR.}\mathsf{eval}_s(x) \wedge x' \neq x \middle| s \leftarrow \text{SPR.}\mathsf{gen}(\kappa), x \leftarrow \{0,1\}^n; x' \leftarrow \mathcal{A}(s,x)\right] < \nu(\kappa),$$

for some negligible $\nu = \nu(\kappa)$, and the probability is taken over the choice of $x \in \{0,1\}^n(\kappa)$ and over the internal randomness of SPR.gen and $\mathcal{A}$.

3 Puncturable Digital Signature Schemes

A puncturable digital signature (PDS) scheme [5] is a digital signature scheme with the additional facility to "puncture" the signing key at some arbitrary message, say, m^*. The resulting punctured signing key allows one to sign all messages except m^*. A PDS is said to be *consistent*, if a secret signing key sk and all possible punctured signing keys $\widehat{\mathsf{sk}}_{m^*}$ derived from sk, for every unpunctured message, produce the same signature, deterministically. In this paper, we shall consider only PDS schemes that are consistent, and hence shall omit that qualifier in the sequel.

The security requirement of a PDS scheme is that the (standard) existential unforgeability should hold for the punctured message m^*. Following Bellare et al. [5], we focus on *selective unforgeability*, wherein the adversary must specify the message m^* at which the signing key needs to be punctured ahead of time, i.e., before receiving the public parameters and the verification key. It then receives the punctured signing key $\widehat{\mathsf{sk}}_{m^*}$ (punctured at m^*) and the verification key of the PDS, and the goal of the adversary is produce a forgery on m^*. The formal definition follows.

Definition 4. (Puncturable Digital Signatures). A puncturable digital signature (PDS) scheme consists of the following polynomial-time algorithms PDS $= (\mathsf{keygen}, \mathsf{sign}, \mathsf{ver}, \mathsf{pkeygen}, \mathsf{psign})$ as detailed below:

- $\mathsf{keygen}(\kappa, \ell)$: *The key generation algorithm takes as input the security parameter κ (in unary) and the length of the messages to be signed ℓ, and outputs a signing-verification key pair* $(\mathsf{sk}, \mathsf{vk})$.
- $\mathsf{sign}(\mathsf{sk}, m)$: *The (deterministic) signing algorithm takes as input the (master) signing key* sk*, and a message $m \in \{0,1\}^\ell$, and outputs a signature σ.*
- $\mathsf{ver}(\mathsf{vk}, (m, \sigma))$: *The verification algorithm takes as input the verification key* vk*, the message-signature pair (m, σ), and outputs 1 if and only if the signature verifies.*
- $\mathsf{pkeygen}(\mathsf{sk}, m^*)$: *The punctured key generation algorithm takes as input the (master) signing key* sk *and a message $m^* \in \{0,1\}^\ell$, and outputs a "punctured" signing key $\widehat{\mathsf{sk}}_{m^*}$.*
- $\mathsf{psign}(\widehat{\mathsf{sk}}_{m^*}, m)$: *The (deterministic) punctured signing algorithm takes as input a punctured signing key $\widehat{\mathsf{sk}}_{m^*}$, and a message $m \in \{0,1\}^\ell$, to generate a signature σ'.*

We say that a punctured signature scheme PDS is *consistent* if for all $\kappa, \ell \in \mathbb{N}$, $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{keygen}(\kappa, \ell)$, $m^* \in \{0,1\}^\ell$, $\widehat{\mathsf{sk}}_{m^*} \leftarrow \mathsf{pkeygen}(\mathsf{sk}, m^*)$, and $m \in \{0,1\}^\ell \setminus m^*$, it holds that: $\mathsf{sign}(\mathsf{sk}, m) = \mathsf{psign}(\widehat{\mathsf{sk}}_{m^*}, m)$. The "consistency" requirement of the PDS stipulates that both the algorithms sign and psign be *deterministic*.

Punctured (selective) unforgeability under chosen message attacks: The (selective) security of a PDS scheme $\text{PDS} = (\mathsf{keygen}, \mathsf{sign}, \mathsf{ver}, \mathsf{pkeygen}, \mathsf{psign})$ is defined using a game between a challenger $\mathcal{C}$ and a (non-uniform) PPT adversary $\mathcal{A}$. The game, parametrized with κ, consists of the following stages:

1. **Selecting Message for Puncturing:** The adversary outputs a message $m^* \in \{0,1\}^\ell$ (the point to be punctured at).
2. **Key and Punctured-Key Generation:** Then the challenger $\mathcal{C}$ runs $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{keygen}(\kappa, \ell)$, and returns the verification key vk to the adversary $\mathcal{A}$. It also computes $\widehat{\mathsf{sk}}_{m^*} \leftarrow \mathsf{pkeygen}(\mathsf{sk}, m^*)$ and sends it to $\mathcal{A}$.[5]
3. **Forgery:** Finally the adversary $\mathcal{A}$ outputs a purported signature σ^* corresponding to the message m^*.

The advantage of $\mathcal{A}$ in this game is $\mathsf{Adv}^{\text{PDS}}_{\mathcal{A}}(\kappa) := \Pr[\mathsf{ver}(\mathsf{vk}, (m^*, \sigma^*)) = 1]$.

Definition 5. *A punctured signature scheme* PDS *is said to be δ-secure if, for all PPT adversaries $\mathcal{A}$, $\mathsf{Adv}^{\text{PDS}}_{\mathcal{A}}(\kappa) \leq \delta(\kappa)$, where $\delta(\kappa)$ is negligible in κ. We say that* PDS *is sub-exponentially secure if it is 2^{κ^ϵ}-secure, for some $0 < \epsilon < 1$.*

3.1 Construction of a PDS scheme.

Our construction of the PDS relies on the *sole* assumption that one-way functions exist. Below, we summarize the construction of our PDS scheme, and give a high level sketch of the construction.

Intuition behind the construction. The construction follows the paradigm of extending one-time signatures into full-fledged signatures using a tree of pseudorandomly generated key pairs [31, 40, 44]. Each message in the message space is associated with a leaf in this tree, and the key pair at that leaf is used to exclusively sign that message. The signature on a message will also certify the leaf's verification key using a "certificate chain" that follows the path from root to leaf in the tree. Our scheme will rely on a punctured PRF to generate this tree. The signing key punctured at a message m^* will include a punctured PRF key, *punctured at all the points in the path from root to the leaf corresponding to m^**; also it will include a small set of certificates that, for every message $m \neq m^*$, can be used to certify the verification key for the first node that is in the path from the root to the leaf corresponding to m, but not in the path from the root to the leaf corresponding to m^*. Compared to the certificate chains used in the standard signature construction, it is important in our case to *verifiably tie* the verification keys to specific nodes in the tree, because otherwise the signer with a punctured signing key can use keys for one leaf to sign the message associated with another leaf.

The construction. Given a pPRF scheme $F = (F.\mathsf{puncture}, F.\mathsf{eval})$ with key space $\mathcal{K}$ and punctured key space $\widehat{\mathcal{K}}$, and a (one-time) digital signature scheme $\Sigma = (\mathsf{keygen}, \mathsf{sign}, \mathsf{verify})$

[5] Note that the adversary does not need access to the signing oracle. This is because the adversary can itself compute $\mathsf{psign}(\widehat{\mathsf{sk}}_{m^*}, m)$ on all messages of its choice other than m^*, and by the "consistency" condition, for such queries, access to $\mathsf{sign}(\mathsf{sk}, \cdot)$ and access to $\mathsf{psign}(\widehat{\mathsf{sk}}_{m^*}, \cdot)$ are equivalent.

(over appropriate domains), we shall construct a PDS scheme $\text{PDS} = (\mathsf{keygen}, \mathsf{sign}, \mathsf{ver}, \mathsf{pkeygen}, \mathsf{psign})$. We use the convention that $\Sigma.\mathsf{keygen}$ accepts its randomness as an input.

We let $\mathcal{M} = \{0,1\}^\ell$ be the message space of our PDS scheme. Let $\mathcal{M}^+$ denote the set of all bit strings of length ℓ or less (including the empty string ϵ). The input domain of the PPRF F is identified with $\mathcal{M}^+$. We write $u \prec z$ (or, $u \preceq z$) to denote that u is a strict prefix (or, respectively, prefix) of z; also let $\mathsf{prefix}(m) = \{u \mid u \preceq m\}$.

It will be convenient to have a notation for "certificate chains." Given $z \in \mathcal{M}^+$ and signing and verification keys of Σ, $(\mathsf{sk}_u, \mathsf{vk}_u)$ for every $u \preceq z$, we define $\mathsf{chain}(z \mid \{\mathsf{sk}_u, \mathsf{vk}_{u0}, \mathsf{vk}_{u1}\}_{u \prec z}) := \{(\mu_u, \sigma_u)\}_{u \prec z}$ where $\mu_u = (\mathsf{vk}_{u0}, \mathsf{vk}_{u1}, u)$ and $\sigma_u = \Sigma.\mathsf{sign}(\mathsf{sk}_u, \mu_u)$. We shall denote a purported chain as $\mathsf{chain}(z \mid \{\mathsf{vk}_u\}_{\epsilon \prec u \prec z})$, when it has the form $\{(\mu_u, \sigma_u)\}_{u \prec z}$ where $\mu_u = (\mathsf{vk}_{u0}, \mathsf{vk}_{u1}, u)$ (for *some* $\{\sigma_u\}_{u \prec z}$). We say that a purported chain $\mathsf{chain}(z \mid \{\mathsf{vk}_u\}_{\epsilon \prec u \prec z})$ *verifies w.r.t.* vk if
$\Sigma.\mathsf{verify}(\mathsf{vk}, \mu_\epsilon, \sigma_\epsilon) = 1$, and for each non-empty $u \prec z$, $\Sigma.\mathsf{verify}(\mathsf{vk}_u, \mu_u, \sigma_u) = 1$.

- $\mathsf{keygen}(\kappa, \ell)$*:* Pick $K \leftarrow \mathcal{K}$. Let $r_\epsilon = F(K, \epsilon)$, and $(\mathsf{sk}_\epsilon, \mathsf{vk}_\epsilon) \leftarrow \Sigma.\mathsf{keygen}(r_\epsilon)$. Output $(\mathsf{sk}, \mathsf{vk})$ where $\mathsf{sk} = (K, \ell)$ and $\mathsf{vk} = (\mathsf{vk}_\epsilon, \ell)$.
- $\mathsf{sign}(\mathsf{sk} = (K, \ell), m \in \{0,1\}^\ell)$*:* For each $u \preceq m$, let $r_u = F(K, u)$, and $(\mathsf{sk}_u, \mathsf{vk}_u) \leftarrow \Sigma.\mathsf{keygen}(r_u)$. Output the tuple $(\Sigma.\mathsf{sign}(\mathsf{sk}_m, m), \mathsf{chain}(m \mid \{\mathsf{sk}_u, \mathsf{vk}_{u0}, \mathsf{vk}_{u1}\}_{u \prec m}))$.
- $\mathsf{ver}(\mathsf{vk}, (m, \sigma))$*:* Parse σ as $(\sigma, \mathsf{chain}(m \mid \{\mathsf{vk}_u\}_{\epsilon \prec u \prec m}))$. Output 1 iff $\Sigma.\mathsf{verify}(\mathsf{vk}_m, m, \sigma_m) = 1$ and the chain verifies with respect to vk.
- $\mathsf{pkeygen}(\mathsf{sk} = (K, \ell), m^* \in \{0,1\}^\ell)$*:* Let $T = \mathsf{prefix}(m^*)$, and $\widehat{K}_T \leftarrow F.F.\mathsf{puncture}(K, T)$. Output $\widehat{\mathsf{sk}}_{m^*} := (\widehat{K}_T, \mathsf{chain}(m^* \mid \{\mathsf{sk}_u, \mathsf{vk}_{u0}, \mathsf{vk}_{u1}\}_{u \prec m^*})$.
- $\mathsf{psign}(\widehat{\mathsf{sk}}_{m^*}, m)$*:* If $m = m^*$, output $\perp$. Else, let $z \preceq m$ be such that $z \in \{v0, v1\}$, where $v \prec m^*$ but $z \not\preceq m^*$. Note that $\mathsf{chain}(z)$ is a prefix of $\mathsf{chain}(m^*)$, which is part of $\widehat{\mathsf{sk}}_{m^*}$. For all u such that $z \preceq u \prec m$,[6] compute $r_u = F.F.\mathsf{eval}(\widehat{K}_T, u)$, and $(\mathsf{sk}_u, \mathsf{vk}_u) = \Sigma.\mathsf{keygen}(r_u)$. Construct $\mathsf{chain}(m)$ using $\mathsf{chain}(z)$ and the above values. Also, compute $\sigma = \Sigma.\mathsf{sign}(\mathsf{sk}_m, m)$. Output $(\sigma, \mathsf{chain}(m))$.

Theorem 1. *Let F be a selectively secure puncturable PRF and Σ be a secure (one-time) digital signature scheme. Then the construction of* PDS *shown above is a secure puncturable digital signature scheme. In particular, we will show that:*

$$\mathsf{Adv}^{\text{PDS}}_{\mathcal{A}}(\kappa) \leq \epsilon_F + (\ell + 1)\epsilon_\Sigma,$$

where ϵ_F and ϵ_Σ are upper bounds on the advantage of PPT adversaries for F and Σ respectively.

Proof sketch: Firstly, note that the construction ensures that signatures constructed using the signature key and the punctured signature keys (on non-punctured points) will always verify.

[6] This would be an empty set if m is the sibling of m^*: i.e., if $m = m^* \oplus 0^{\ell-1}1$; in this case $z = m$ and $\mathsf{chain}(z) = \mathsf{chain}(m^*)$

In the experiment of pPRF security, let $T = \mathsf{prefix}(m^*)$. First, we consider a hybrid experiment in which the challenger uses the pPRF key K only for sampling r_u for $u \notin T$, and it uses independently sampled random values r_u for $u \in T$. The difference in the probability of $\mathcal{A}$ winning in the original and modified experiments is at most ϵ_F (*selective* security of F is sufficient for this).

Next, we argue that the advantage of $\mathcal{A}$ in the modified experiment is at most $(\ell+1)\epsilon_\Sigma$. For this we consider an adversary $\mathcal{A}_\Sigma$ which accepts vk^Σ from outside and access to a one-time signature oracle using the corresponding signing key. It internally simulates the above experiment perfectly, with the following further modification: it uniformly randomly chooses $u^* \leftarrow T$ and lets $\mathsf{vk}_{u^*} = \mathsf{vk}^\Sigma$. Also, instead of using sk_{u^*} (which it does not have access to), it queries the signing oracle for a signature on μ_{u^*}, where for ease of notation we have defined

$$\mu_u = \begin{cases} (u, \mathsf{vk}_{u0}, \mathsf{vk}_{u1}) & \text{if } u \prec m^* \\ u & \text{if } u = m^*. \end{cases} \tag{1}$$

Note that when vk^Σ is honestly generated, this is indeed a perfect simulation of the hybrid experiment above.

Now, suppose $\mathcal{A}$ successfully produces a forgery – i.e., outputs $(\sigma^*, \mathsf{chain}(m^* \mid \{\mathsf{vk}'_u\}_{\epsilon \prec u \prec m^*}))$ where $\Sigma.\mathsf{verify}(\mathsf{vk}_m, m, \sigma^*) = 1$ and the chain verifies w.r.t. vk. Let $\mathsf{vk}'_\epsilon = \mathsf{vk}_\epsilon$. Also, we define μ'_u analogous to μ_u in (1), using vk'_{ub} instead of vk_{ub}. We consider two cases: either $\mathsf{vk}'_m = \mathsf{vk}_m$ or $\exists v^* \prec m^*$ such that $\mathsf{vk}'_{v^*} = \mathsf{vk}_{v^*}$ and $\mathsf{vk}'_{v^*b} \neq \mathsf{vk}_{v^*b}$, where $b \in \{0,1\}$ and $v^*b \preceq m^*$. In the former case, we define $v^* := m^*$. Then, in either case $\mu'_{v^*} \neq \mu_{v^*}$. Further, with probability $1/(\ell+1)$ $v^* = u^*$ (independent of everything else). When this happens, we have obtained a Σ-signature on μ'_{u^*} which verifies w.r.t. the key $\mathsf{vk}_{u^*} = \mathsf{vk}^\Sigma$, where as we queried the corresponding signing oracle on a single different message, μ_{u^*}. Since this probability is upper bounded by ϵ_Σ, the probability $\mathcal{A}$ succeeds in the modified experiment is upper bounded by $(\ell+1)\epsilon_\Sigma$. □

4 Witness Maps

In this section we formally define the new primitives called Compact Witness Maps and Dual Mode Witness Maps.

Recall that $R \subseteq \{0,1\}^* \times \{0,1\}^*$ is said to be an **NP** relation if membership in it can be computed in time polynomial in the length of the first input.

Given an **NP** relation R, we define the **NP** language $L_R := \{x \mid \exists w, (x,w) \in R\}$. When referring to $(x,w) \in R$, where R is a given **NP** relation, x is called the statement and w the witness. It will be convenient for us to consider **NP** relations parametrized with their input length: Below we let $R_\ell := R \cap \{0,1\}^\ell \times \{0,1\}^*$.

Definition 6 (Compact Witness Map (CWM)). *For $\alpha \geq 0$, an α-CWM for an* **NP** *relation R is a triple* $\mathrm{CWM} = (\mathsf{setup}, \mathsf{map}, \mathsf{check})$ *where* setup *is a PPT algorithm and the other two are deterministic polynomial time algorithms such that:*

- $\mathsf{setup}(\kappa, \ell)$ *outputs a string* K *of length polynomial in the security parameter* κ *and* ℓ*.*
- *Completeness: For any polynomial* ℓ*,* $\forall (x, w) \in R_{\ell(\kappa)}$*,* $\forall \mathsf{K} \leftarrow \mathsf{setup}(\kappa, \ell(\kappa))$*,*

$$\mathsf{check}(\mathsf{K}, x, \mathsf{map}(\mathsf{K}, x, w)) = 1.$$

- *Lossiness: For any polynomial* ℓ*,* $\forall \mathsf{K} \leftarrow \mathsf{setup}(\kappa, \ell(\kappa))$*,* $\forall x \in \{0,1\}^{\ell(\kappa)}$*,*

$$|\{\mathsf{map}(\mathsf{K}, x, w) \mid (x, w) \in R_{\ell(\kappa)}\}| \leq 2^{\alpha}.$$

- *Soundess: For any polynomial* ℓ *and any PPT adversary* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathrm{CWM}}_{\mathcal{A}}(\kappa)$ *defined below is negligible:*

$$\Pr_{\mathsf{K} \leftarrow \mathsf{setup}(\kappa, \ell(\kappa))}[\mathcal{A}(\mathsf{K}) \to (x^*, w^*), \mathsf{check}(\mathsf{K}, x^*, w^*) = 1, x^* \notin L_R].$$

A 0-CWM is called a Unique Witness Map *(UWM).*

The above definition has perfect security in the sense that the completeness and lossiness conditions hold for *every possible* K that CWM.setup can output with positive probability. A statistical version, where this needs to hold with all but negligible probability over the choice of K will suffice for all our applications. But for simplicity, we shall use the perfect version above. It is useful to consider a variant of the definition with a *selective soundness* guarantee, in which the adversary is required to generate x^* first (given κ, ℓ) before it gets K. For some applications (e.g., construction of a witness encryption scheme from a UWM) this level of soundness suffices. It also provides an intermediate target for constructions, as one can convert a selectively sound CWM to a standard CWM by relying on complexity leveraging (as we shall do in our construction in Section 4.1).

Definition 7 (Dual Mode Witness Maps (DMWM)). *An* α*-DMWM with tag space* $\mathcal{T}$ *for an* **NP** *relation* R *is a tuple* $\mathrm{DMWM} = (\mathsf{setup}, \mathsf{map}, \mathsf{check}, \mathsf{extract})$ *where* setup *is a PPT algorithm and the others are deterministic polynomial time algorithms such that:*

- $\mathsf{setup}(\kappa, \ell, \mathsf{tag})$ *outputs* $(\mathsf{K}, \mathsf{td})$*, where* κ *is a security parameter,* $\ell(\kappa)$ *is a polynomial, and* $\mathsf{tag} \in \mathcal{T}$*,* K *and* td *are strings of length polynomial in* κ*.*
- *Completeness:* $\forall \mathsf{tag}, \mathsf{tag}' \in \mathcal{T}$ *for all polynomials* ℓ*,* $\forall (x, w) \in R_{\ell(\kappa)}$*,* $\forall \mathsf{K} \leftarrow \mathsf{setup}(\kappa, \ell(\kappa), \mathsf{tag})$*,*

$$\mathsf{check}(\mathsf{K}, \mathsf{tag}', x, \mathsf{map}(\mathsf{K}, \mathsf{tag}', x, w)) = 1.$$

- *Hidden Tag: For any PPT adversary* $\mathcal{A}$*,* $\mathsf{Adv}^{\text{DMWM-hide}}_{\mathcal{A}}(\kappa)$ *defined below is negligible:*

$$\Big|\Pr\big[\mathcal{A}(\kappa, \ell) \to (\mathsf{tag}_0, \mathsf{tag}_1, \mathsf{st}), b \leftarrow \{0,1\}, (\mathsf{K}, \mathsf{td}) \leftarrow \mathsf{setup}(\kappa, \ell(\kappa), \mathsf{tag}_b), \mathcal{A}(\mathsf{K}, \mathsf{st}) \to b', b = b'\big] - \frac{1}{2}\Big|.$$

- *Extraction: For any polynomial ℓ, for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathrm{DMWM}}_{\mathcal{A}}(\kappa)$ defined below is negligible:*

$$\mathsf{Adv}^{\mathrm{DMWM}}_{\mathcal{A}}(\kappa) := \Pr[\mathcal{A}(\kappa,\ell) \to (\mathsf{tag},\mathsf{st}), (\mathsf{K},\mathsf{td}) \leftarrow \mathsf{setup}(\kappa,\ell(\kappa),\mathsf{tag}), \mathcal{A}(\mathsf{K},\mathsf{st}) \to (x^*,w^*), \\ \mathsf{check}(\mathsf{K},\mathsf{tag},x^*,w^*) = 1, (x^*,\mathsf{extract}(\mathsf{td},x,w^*)) \notin R_{\ell(\kappa)}]$$

- *Cumulative Lossiness: $\forall \mathsf{tag}, \ell, \forall \mathsf{K} \leftarrow \mathsf{setup}(\kappa,\ell,\mathsf{tag}), \forall x \in L_{R_\ell}$, there exist (inefficient) functions $\mathsf{compress}_{\mathsf{K},x} : \{0,1\}^* \to S_{\mathsf{K},x}$ and $\mathsf{expand}_{\mathsf{K},x} : S_{\mathsf{K},x} \times \{0,1\}^* \to \{0,1\}^*$ such that $|S_{\mathsf{K},x}| \leq 2^{\alpha(\kappa)}$, and for all $\mathsf{tag}' \neq \mathsf{tag}$,*

$$\mathsf{map}(\mathsf{K},\mathsf{tag}',x,w) = \mathsf{expand}_{\mathsf{K},x}(\mathsf{compress}_{\mathsf{K},x}(w),\mathsf{tag}').$$

4.1 Unique Witness Maps

In this section, we present a construction of 0-CWM or an UWM.

4.1.1 A UWM for any NP Relation. Now we present the construction of our UWM system UWM for any **NP** relation R (see Figure 1). The main building blocks of our construction are a punctured digital signature (PDS) scheme PDS and an $i\mathcal{O}$ scheme (denoted as $i\mathcal{O}$).

Let PDS $= (\mathsf{keygen},\mathsf{sign},\mathsf{ver},\mathsf{pkeygen},\mathsf{psign})$ be a secure punctured digital signature scheme and $i\mathcal{O}$ be a secure indistinguishability obfuscator for circuits.

1. UWM.$\mathsf{setup}(\ell,\kappa)$: Generate $(\mathsf{sk},\mathsf{vk}) \leftarrow$ PDS.$\mathsf{keygen}(\ell,\kappa)$. Then create an obfuscated program $P \leftarrow i\mathcal{O}(\mathsf{Endorse}^{R_\ell}_{\mathsf{sk}})$, where the program $\mathsf{Endorse}^{R_\ell}_{\mathsf{sk}}$ is as shown below. Output $\mathsf{K} = (\mathsf{vk}, P)$.
2. UWM.$\mathsf{map}(\mathsf{K},x,w)$: Parse K as (vk,P). Output $w^* \leftarrow P(x,w)$.
3. UWM.$\mathsf{check}(\mathsf{K},x,w^*)$: Parse K as (vk,P). Output PDS.$\mathsf{ver}(\mathsf{vk},x,w^*)$.

Program $\mathsf{Endorse}^{R_\ell}_{\mathsf{sk}}((x,w))$
Constant: Signing key sk
Input Domain: $(x,w) \in \{0,1\}^\ell \times \{0,1\}^{\ell'}$
if $(x,w) \in R_\ell$ **then**
output PDS.$\mathsf{sign}(\mathsf{sk},x)$
else
output $\perp$

Program $\mathsf{pEndorse}^{R_\ell}_{\widehat{\mathsf{sk}}_{x^*}}((x,w))$
Constant: Punctured signing key $\widehat{\mathsf{sk}}_{x^*}$
Input Domain: $(x,w) \in \{0,1\}^\ell \times \{0,1\}^{\ell'}$
if $(x,w) \in R_\ell$ and $x \neq x^*$ **then**
output PDS.$\mathsf{psign}(\widehat{\mathsf{sk}}_{x^*},x)$
else
output $\perp$

Fig. 1. The UWM for an **NP** relation R. The program $\mathsf{pEndorse}^{R_\ell}_{\widehat{\mathsf{sk}}_{x^*}}$ is used only in the proof.

Theorem 2. *Let $i\mathcal{O}$ be a (polynomially) secure indistinguishability obfuscator for circuits and* PDS *be a (polynomially) secure consistent puncturable digital signature scheme. Then* UWM *defined in Figure 1 is a UWM for the* **NP** *relation R satisfying selective soundness.*

Proof. Firstly, we note that UWM satisfies perfect completeness (assuming $i\mathcal{O}$ and PDS are perfectly correct). Also, it satisfies uniqueness, since (x, w) is deterministically mapped to the signature on x, independent of w. Below, we shall prove that the scheme is sound as well.

Consider an adversary $\mathcal{A}$ in the definition of $\mathsf{Adv}^{\mathrm{UWM}}_{\mathcal{A}}(\kappa)$. Note that $\mathcal{A}$ outputs a point x^* first. We consider a hybrid experiment where, after $\mathcal{A}$ outputs x^*, K is derived from a modified UWM.setup: The modified UWM.setup is only different in that instead of using $\mathsf{Endorse}^{R_\ell}_{\mathsf{sk}}$, the program $\mathsf{pEndorse}^{R_\ell}_{\widehat{\mathsf{sk}}_{x^*}}$ (also shown in Figure 1) is used, where $\widehat{\mathsf{sk}}_{x^*} \leftarrow \mathrm{PDS}.\mathsf{pkeygen}(\mathsf{sk}, x^*)$.

We claim that the advantage $\mathcal{A}$ has in the modified experiment can only be negligibly more than that in the original experiment. For this consider, a coupled execution of the two experiments, with $\mathcal{A}$'s random tape being the same in the two executions. Then it is enough to upper bound the the difference of probabilities of the condition $\mathrm{UWM}.\mathsf{check}(\mathsf{K}, x^*, w^*) = 1 \ \wedge \ x^* \notin L_{R_\ell}$ holding in the modified experiment and in the original experiment. Fix a choice of randomness that maximizes this difference, δ. We shall describe a (non-uniform) adversary $\mathcal{A}_{i\mathcal{O}}$, which internally runs the coupled experiment with this choice of randomness for $\mathcal{A}$. Let x^* be the output of $\mathcal{A}$ with this choice. Note that for $\delta > 0$, we need $x^* \notin L_{R_\ell}$. For such x^*, observe that $\mathsf{Endorse}^{R_\ell}_{\mathsf{sk}}$ and $\mathsf{pEndorse}^{R_\ell}_{\widehat{\mathsf{sk}}_{x^*}}$ are functionally equivalent programs (for all sk). This is because, if $(x, w) \in R_\ell$, then $x \neq x^*$ and the consistency of the PDS scheme guarantees that $\mathrm{PDS}.\mathsf{sign}(\mathsf{sk}, x) = \mathrm{PDS}.\mathsf{sign}(\widehat{\mathsf{sk}}_{x^*}, x)$. So $\mathcal{A}_{i\mathcal{O}}$ can output the pair of programs $\mathsf{Endorse}^{R_\ell}_{\mathsf{sk}}$ and $\mathsf{pEndorse}^{R_\ell}_{\widehat{\mathsf{sk}}_{x^*}}$. It receives back an obfuscated program P and carries out the rest of the UWM security game with $\mathcal{A}$ using P. If $P \leftarrow i\mathcal{O}(\mathsf{Endorse}^{R_\ell}_{\mathsf{sk}})$, then this game is exactly the original game, and otherwise it is the modified game. Hence, $\mathcal{A}_{i\mathcal{O}}$ distinguishes between these two cases with advantage δ. Hence, by the security of $i\mathcal{O}$, δ is negligible; this in turn shows that the advantage $\mathcal{A}$ has in the modified experiment is only negligibly far from that in the original experiment.

Next, we argue that in the modified selective soundness experiment $\mathcal{A}$ has negligible advantage. Note that in the modified experiment, $\mathcal{A}$ outputs a string $x^* \in \{0,1\}^\ell$, gets back $(\mathsf{vk}, \widehat{P})$, where $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathrm{PDS}.\mathsf{keygen}(\ell, \kappa)$, and P is generated from the punctured secret-key $\widehat{\mathsf{sk}}_{x^*}$, outputs a purported signature w^*, and wins if $\mathrm{PDS}.\mathsf{ver}(\mathsf{vk}, x^*, w^*) = 1$. By the security of PDS, the probability of $\mathcal{A}$ winning is at most $\mathsf{Adv}^{\mathrm{PDS}}_{\mathcal{A}}(\kappa)$, which is negligible. □

Remark 1. In the above proof, we only show selective soundness of UWM. We note that, one can transform a selectively sound UWM to an adaptively sound one using complexity leveraging, when appropriate. This can be done by choosing PDS to be $2^{-(\ell+\kappa)}$-secure punctured digital signature scheme and $i\mathcal{O}$ to be $2^{-(\ell+\kappa)}$-secure indistinguishability obfuscator for circuits respectively (i.e., the advantages $\mathsf{Adv}^{\mathrm{PDS}}_{\mathcal{A}}(\kappa_1) \leq 2^{-(\ell+\kappa)}$ and $\mathsf{Adv}^{i\mathcal{O}}_{Samp,\mathcal{D}}(\kappa_2) \leq 2^{-(\ell+\kappa)}$, where κ_1 and κ_2 are the security parameters for PDS and $i\mathcal{O}$ respectively, and κ is the security parameter for UWM). One can set κ_1 and κ_2 to be large enough to satisfy this.

4.1.2 Implication to Witness Encryption. In this section, we show that a full-fledged UWM implies Witness encryption (WE). We start by recalling the definition of WE from [25].

Definition 8 (Witness Encryption). *A witness encryption scheme* WE *for an* **NP** *language L_R with witness relation R consists of the following polynomial-time algorithms:*

- **Encrypt**$(1^\kappa, x, M)$. *The encryption algorithm takes as input the security parameter 1^κ, an unbounded-string $x \in \{0,1\}^*$, and a message $M \in \{0,1\}$, and outputs a ciphertext C.*
- **Decrypt**(w, C). *The decryption algorithm takes as input a ciphertext C, and an unbounded length string $w \in \{0,1\}^*$, and outputs a message M or the symbol $\perp$.*

The algorithms must satisfy the following requirements:

- *Completeness: $\forall \kappa \in \mathbb{N}$, $\forall (x,w) \in R_\ell$, and any message $M \in \{0,1\}$, we have:* **Decrypt**$(w,$ **Encrypt**$(1^\kappa, x, M) = M$.
- *Soundness Security: For any PPT adversary $\mathcal{A}_{\text{WE}}$, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for any $x \notin L_R$ we have:*

$$\Big|\Pr[\mathcal{A}_{\text{WE}}(\textbf{Encrypt}(1^\kappa, x, 0)) = 1] - \Pr[\mathcal{A}_{\text{WE}}(\textbf{Encrypt}(1^\kappa, x, 1)) = 1]\Big| < \mathsf{negl}(\kappa).$$

In the following, we will also need to use a generalized version of the Goldreich-Levin (GL) theorem, as stated below.

Lemma 3 (Generalized Goldreich-Levin Theorem). *There exists a PPT inverter $\mathcal{A}'$ and a non-zero polynomial $q(\cdot)$ such that, for any machine $\mathcal{A}$ and any $(\alpha, \beta) \in \{0,1\}^k \times \{0,1\}^\ell$ such that $p(\alpha) := \Pr[\mathcal{A}(\alpha, r) = \langle \beta, r\rangle : r \xleftarrow{\$} \{0,1\}^\ell]$ (where $\langle \cdot, \cdot \rangle$ denotes the inner product over the binary field), then $\Pr[\mathcal{A}'^{\mathcal{A}(\alpha,\cdot)}(1^\ell, \alpha) = \beta] \geq q(p(\alpha) - \frac{1}{2})$.*

We now proceed to give the construction of the witness encryption scheme WE from a unique witness map UWM (see Figure 2). Before giving the construction, we give a high-level idea behind the construction.

Intuition behind the construction. We show a construction of WE for an arbitrary **NP** language L starting from an UWM for the language $L_{OR} = L \vee L'$, where L' is another **NP** language whose YES instances are *indistinguishable* from NO instances. To WE encrypt a bit $m \in \{0,1\}$ with respect to an **NP** statement $x \in L$, we sample an YES instance from the **NP** language L'. We do so by sampling a pseudo-random string $y = G(z)$, such that z serves as a valid witness corresponding to the string y. We then consider the language $L_{OR} = L \vee L'$ which consists of instances $\hat{x}$ of the form "either $x \in L \vee y$ is pseudorandom". We use the UWM to derive a representative witness w^* for a statement corresponding to this augmented **NP** language (using witness z) and then derive the Goldreich-Levin hardcore bit of w^* to be used as a one-time pad to encrypt the bit m. The decryptor can derive the same representative witness w^* using

Let R_1 be an **NP** relation, and L_{R_1} be the corresponding **NP** language defined as $L_{R_1} := \{x \mid \exists w, (x,w) \in R_1\}$. Let R_2 be another **NP** relation defined as $(y,z) \in R_2$ if and only if $y = G(z)$, where $G : \{0,1\}^\kappa \to \{0,1\}^{q(\kappa)}$ is a pseudo-random generator, and $q(\cdot)$ is an arbitrary polynomial. Also, let L_{R_2} be the corresponding **NP** language defined as $L_{R_2} := \{y \mid \exists z, (y,z) \in R_2\}$. Further, we assume that R_1 and R_2 are parameterized with their input lengths. Define the following derived **NP** relation R_{OR} and language L_{OR} as:

$$R_{OR}\big((x,y),(w,z)\big) = 1 \text{ iff } R_1(x,w) = 1 \ \vee \ R_2(y,z) = 1, \text{ and}$$
$$L_{OR} = \{x,y\} \mid \exists (w,z), (x,w) \in R_1 \ \vee \ (y,z) \in R_2.$$

Note that, the relation R_{OR} is parameterized with the input length $\ell' = \max\{|\ x\ |, |\ y\ |\}$.

(a) Let $\text{UWM} = (\text{UWM.}\mathsf{setup}, \text{UWM.}\mathsf{map}, \text{UWM.}\mathsf{check})$ be a (selectively) sound unique witness map (UWM) (refer to Section 4.1) for the language L_{OR}. Further, let the length of the representative w^* of UWM be $p(\kappa)$ bits, for some polynomial $p(\cdot)$.

(b) Let $\text{GL}(\pi, r)$ denote the Goldreich-Levin (GL) hardcore bit [28] of π using randomness r. Recall that, the GL predicate is the bit-wise inner product of π and r.

1. **Encrypt**$(1^\kappa, x, M \in \{0,1\})$: Takes as input an instance $x \in L_{R_1}$, and a message $M \in \{0,1\}$. Do the following:
   - Sample $z \leftarrow \{0,1\}^\kappa$ and $r \leftarrow \{0,1\}^{p(\kappa)}$ uniformly at random, and compute $y = G(z)$.
   - Run $\mathsf{K} \leftarrow \text{UWM.}\mathsf{setup}(\ell', \kappa)$.
   - Generate a representative witness $w^* = \mathsf{map}\big(\mathsf{K}, (x,y), (\perp, z)\big)$, using $(\perp, z)$ as witness. Note that, the statement $(x,y) \in L_{OR}$.
   - Compute the GL hardcore bit $b = \text{GL}(w^*, r)$ and compute $c = b \oplus M$.
   - Output the ciphertext $C = (\mathsf{K}, y, r, c)$.
2. **Decrypt**$(1^\kappa, C, w)$: On input a ciphertext $C = (\mathsf{K}, y, r, c)$, and a witness w satisfying $(x,w) \in L_{R_1}$, do the following:
   - Compute the representative witness $w^* = \mathsf{map}\big(\mathsf{K}, (x,y), (w, \perp)\big)$. Note that, the witness $(w, \perp)$ is also consistent with the statement (x,y), and hence $(x,y) \in L_{OR}$.
   - Compute $b = \text{GL}(w^*, r)$ and recover the message $M = c \oplus b$.

Fig. 2. Construction of Witness Encryption Scheme WE from UWM

his witness for $x \in L$ (which is also a valid witness for L_{OR}) and therefore decrypt. Intuitively, if an adversary can break WE security, then it can distinguish encryptions of 0 and 1 with non-negligible probability even if x is a false statement. This means that, using Goldreich-Levin decoding, it can compute the correct value w^* given y with non-negligible probability. Furthermore this value w^* is a valid representative witness for the statement $\hat{x}$. At this point, we switch the YES instance of L' to a NO instance (this can be done by sampling a random y, instead of a pseudorandom y), without affecting the advantage of the adversary much. Hence, it must also compute a valid representative witness for $\hat{x}$ if we switch y to false. But this contradicts the soundness of UWM. We remark that, for this reduction it suffices even if the UWM is only *selectively sound*.

Theorem 3. *If* UWM *is a selectively sound UWM for the* **NP** *relation* R_{OR}*, then* WE *defined in Figure 2 is a WE for the* **NP** *relation* R_1*.*

Proof. We show that any adversary $\mathcal{A}_{\text{WE}}$ breaking the soundness security of WE with a noticeable advantage can be transformed into an adversary $\mathcal{A}_{\text{UWM}}$ breaking the selective soundness of UWM. At first, we show that the adversary $\mathcal{A}_{\text{WE}}$ breaking the soundness security of WE can be converted into a predictor $\mathcal{A}'$ for the generalized Goldreich-Levin theorem. In more details, let the adversary $\mathcal{A}_{\text{WE}}$ can predict the bit M with non-negligible probability on some instance $x \notin L_{R_1}$. Also, let $\alpha = (\mathsf{K}, y)$, and $\beta = w^*$ from the generalized GL theorem. This implies that we can construct a distinguisher $\mathcal{A}$ that on input $\big(\alpha = (\mathsf{K}, y), r\big)$ can distinguish the bit b from a random bit with non-negligible probability. Hence, by Lemma 3, we can use this distinguisher $\mathcal{A}$ to construct a predictor $\mathcal{A}'$, who given $\alpha = (\mathsf{K}, y)$ can predict the pre-image w^* with non-negligible probability. This implies that the predictor outputs w^* such that $w^* = \mathsf{map}\big(\mathsf{K}, (x, y), (\perp, z)\big)$ with non-negligible probability. At this point, instead of computing $y = G(z)$, we sample a random $y \leftarrow \{0,1\}^{p(\kappa)}$. The security of the PRG G ensures that this switch is indistinguishable to $\mathcal{A}'$. Hence the probability that $\mathcal{A}'$ outputs a "valid" w^* (w^* such that $\mathsf{check}(\mathsf{K}, (x, y), w^*) = 1$) continues to hold, except with a negligible probability. However, note that, with very high probability it holds that $(x, y) \notin L_{OR}$. This contradicts the soundness property of UWM, since the adversary $\mathcal{A}'$ outputs a valid representative witness corresponding to a false statement $(x, y) \notin L_{OR}$. □

4.2 Cumulative Lossy and All-Lossy-But-One Trapdoor Functions

4.2.1 Cumulative Lossy Trapdoor Functions. In this section, we introduce the notion of "cumulative" lossy trapdoor functions (C-LTDF). A (standard) lossy trapdoor function (LTDF) f can be sampled in one of two *indistinguishable* modes – *injective* or *lossy*. In the injective mode, the function f can be efficiently inverted with the knowledge of a trapdoor; whereas in the lossy mode the function statistically loses a lot of information about its input. We say that a function f with domain $\{0,1\}^n$ is (n, k)-lossy if its image size is at most 2^{n-k}. This implies that mapping a random x to $f(x)$ loses, on average, at least k bits of information about x.

Now, consider the information about x revealed by $(f_1(x), \cdots, f_m(x))$, where $f_1, \cdots, f_m$ are m independently sampled functions from an (n, k)-lossy function family. According to the current definitions and constructions of LTDFs, up to $m(n-k)$ bits could be revealed about x; if $m \geq n/(n-k)$, x could be completely determined by these values.

This is where C-LTDF differs from an LTDF. In a C-LTDF, the amount of information about x that $(f_1(x), \cdots, f_m(x))$ reveals is bounded by a *cumulative loss parameter* α, *irrespective of how large* m *is*. Here the lossy functions f_i can all be sampled independently, but using the same public parameters.

We now formally define C-LTDF and the corresponding properties associated with it.

Definition 9 (C-LTDF). *Let $\kappa \in \mathbb{N}$ be the security parameter, and $\ell, \alpha : \mathbb{N} \to \mathbb{N}$. A* (ℓ, α)-cumulative lossy trapdoor function family (C-LTDF) *is a tuple of (probabilistic) polynomial time algorithms* $(\mathsf{setup}, \mathsf{sample}_{\mathsf{inj}}, \mathsf{sample}_{\mathsf{loss}}, \mathsf{eval}, \mathsf{invert})$ *(the last two being deterministic), having properties as follows:*

- **Parameter Generation.** *The setup algorithm* $\mathsf{setup}(\kappa)$ *outputs a public parameter* pp.
- **Sampling: Injective mode.** *The algorithm* $\mathsf{sample}_{\mathsf{inj}}(\kappa, \mathsf{pp})$ *outputs the tuple* $(\mathsf{ek}, \mathsf{tk})$ *such that* $\mathsf{invert}(\mathsf{tk}, \mathsf{eval}(\mathsf{ek}, x)) = x$ *for all* $x \in \{0,1\}^{\ell(\kappa)}$ *(i.e.,* $\mathsf{eval}(\mathsf{ek}, \cdot)$ *computes an injective function* $f_{\mathsf{ek}}(\cdot)$ *and* $\mathsf{invert}(\mathsf{tk}, \cdot)$ *computes* $f_{\mathsf{ek}}^{-1}(\cdot)$*).*
- **Sampling: Lossy mode.** *For all* pp *in the support of* $\mathsf{setup}(\kappa)$ *there exists an (inefficient) function* $\mathsf{compress}_{\mathsf{pp}} : \{0,1\}^{\ell(\kappa)} \to R_{\mathsf{pp}}$ *with range* $|R_{\mathsf{pp}}| \leq 2^{\ell(\kappa)-\alpha(\kappa)}$*, and for all* ek *in the support of* $\mathsf{sample}_{\mathsf{loss}}(\kappa, \mathsf{pp})$ *there exists an (inefficient) function* $\mathsf{expand}_{\mathsf{ek}}(\cdot)$ *such that the following holds: for all* $x \in \{0,1\}^{\ell(\kappa)}$ *we have* $\mathsf{eval}(\mathsf{ek}, x) = \mathsf{expand}_{\mathsf{ek}}(\mathsf{compress}_{\mathsf{pp}}(x))$.
- **Indistinguishability of modes.** *The ensembles* $\{(\mathsf{pp}, \mathsf{ek}) : \mathsf{pp} \leftarrow \mathsf{setup}(\kappa), (\mathsf{ek}, \mathsf{tk}) \leftarrow \mathsf{sample}_{\mathsf{inj}}(\kappa, \mathsf{pp})\}_{\kappa \in \mathbb{N}}$ *and* $\{(\mathsf{pp}, \mathsf{ek}) : \mathsf{pp} \leftarrow \mathsf{setup}(\kappa), \mathsf{ek} \leftarrow \mathsf{sample}_{\mathsf{loss}}(\kappa, \mathsf{pp})\}_{\kappa \in \mathbb{N}}$ *are computationally indistinguishable.*

4.2.1.1 **A Construction of C-LTDF based on the d-Linear Assumption.** In this section, we present the construction of C-LTDF from the d-linear assumption. In Appendix A we also show how to construct (relaxed) C-LTDF from LWE.

The d-linear assumption [7] is a generalization of the Decision Diffie-Hellman (DDH) assumption. In particular, the 1-Linear assumption is precisely the DDH assumption and the 2-Linear assumption is the Decision Linear assumption [7]. For our construction, we will actually need Matrix d-Linear assumption, which is implied by the d-Linear assumption, as shown by Naor and Segev [43]. Before specifying the assumptions, we will need to introduce some additional notations as follows.

Additional Notation. Let $\mathsf{GroupGen}$ be a PPT algorithm that takes as input the security parameter κ and outputs the a triplet $(\mathbb{G}, p, g)$ where $\mathbb{G}$ is a group of prime order p generated by $g \in \mathbb{G}$. We denote by $\mathsf{Rk}_i(\mathbb{F}_p^{a \times b})$ the set of all $a \times b$ matrices over the field $\mathbb{F}_p$ of rank i. For a vector $\boldsymbol{x} = (x_1, \cdots x_n) \in \mathbb{F}_p^n$, we define $g^{\boldsymbol{x}}$ to be the column vector $(g^{x_1}, \cdots, g^{x_n}) \in \mathbb{G}^n$. If $M = (m_{ij})$ is a $n \times n$ matrix over $\mathbb{F}_p$, we denote by g^M the $n \times n$ matrix over $\mathbb{G}$ given by $(g^{m_{ij}})$. Given any matrix $M = (m_{ij}) \in \mathbb{F}_p^{n \times n}$ and a column vector $\boldsymbol{y} = (y_1, \cdots y_n) \in \mathbb{G}^n$, we define by $\boldsymbol{y}^M = \left(\prod_{j=1}^n y_j^{m_{1j}}, \cdots, \prod_{j=1}^n y_j^{m_{nj}}\right) \in \mathbb{G}^n$. For any matrix $R = (r_{ij}) \in \mathbb{G}^{n \times n}$ and a column vector $\boldsymbol{z} = (z_1, \cdots, z_n) \in \mathbb{F}_p^n$, we define by $R^{\boldsymbol{z}} = \left(\prod_{j=1}^n r_{1j}^{z_j}, \cdots, \prod_{j=1}^n r_{nj}^{z_j}\right) \in \mathbb{G}^n$. This naturally generalizes for two matrices as well. In other words, for two matrices $R \in \mathbb{G}^{n \times n}$ and $Z \in \mathbb{F}_p^{n \times n}$, we denote by $R^Z = (R^{\boldsymbol{z_1}}, \cdots, R^{\boldsymbol{z_n}}) \in \mathbb{G}^{n \times n}$, where each $R^{\boldsymbol{z_i}}$ $(i \in [n])$ is a column vector in $\mathbb{G}^n$ (as defined above) and for all i, $\boldsymbol{z_i}$ denotes the i^{th} column of the matrix Z.

Definition 10 (d-Linear assumption [7]). *Let $d \geq 1$ be an integer, and* $\mathsf{GroupGen}$ *be as above. We say that the* d-linear assumption *holds for* $\mathsf{GroupGen}$ *if the following two distributions are computationally indistinguishable:*

$$\{(g, \mathbb{G}, p, \{g_i, g_i^{r_i}\}_{i=1}^d, h, h^{\sum_{i=1}^d r_i}) : (g, \mathbb{G}, p) \leftarrow \mathsf{GroupGen}; g_i, h \xleftarrow{\$} \mathbb{G}; r_i \xleftarrow{\$} \mathbb{Z}_p\},$$
$$\{(g, \mathbb{G}, p, \{g_i, g_i^{r_i}\}_{i=1}^d, h, h^{r}) : (g, \mathbb{G}, p) \leftarrow \mathsf{GroupGen}; g_i, h \xleftarrow{\$} \mathbb{G}; r_i, r \xleftarrow{\$} \mathbb{Z}_p\},$$

Definition 11 (Matrix d-Linear assumption [43]). *Let a, b, d be integers s.t. $1 \leq d \leq \min\{a, b\}$. Let* GroupGen *be as above. We say that the $(a \times b)$-*Matrix d-linear assumption *holds for* GroupGen *if, for all i, j with $d \leq i \leq j \leq \min a, b$, the two distributions $\{\mathbb{G}, p, g, g^R : (\mathbb{G}, p, g) \leftarrow \mathsf{GroupGen}, R \leftarrow \mathsf{Rk}_i(\mathbb{Z}_p^{a\times b})\}$ and $\{\mathbb{G}, p, g, g^R : (\mathbb{G}, p, g) \leftarrow \mathsf{GroupGen}, R \leftarrow \mathsf{Rk}_j(\mathbb{Z}_p^{a\times b})\}$ are computationally indistinguishable.*

Lemma 4. [43] *If the d-Linear assumption holds for* GroupGen*, then so does the matrix d-Linear assumption.*

The construction. Let $d \geq 1$ be a positive integer. Define the tuple C-LTDF $= (\mathsf{setup}, \mathsf{sample}_{\mathsf{inj}}, \mathsf{sample}_{\mathsf{loss}}, \mathsf{eval}, \mathsf{invert})$ as follows:

1. $\mathsf{setup}(\kappa)$: On input the security parameter κ, do the following:
   - Run $\mathsf{GroupGen}(\kappa)$ to obtain the tuple $(\mathbb{G}, p, g)$.
   - Sample a random matrix $M \xleftarrow{\$} \mathsf{Rk}_d(\mathbb{Z}_p^{n\times n})$ and let $S = g^M \in \mathbb{G}^{n\times n}$.
   - Set the public parameter $\mathsf{pp} = (\mathbb{G}, p, g, S)$.
2. $\mathsf{sample}_{\mathsf{inj}}(\kappa, \mathsf{pp})$: On input pp, chooses a random matrix $M_1 \xleftarrow{\$} \mathsf{Rk}_n(\mathbb{Z}_p^{n\times n})$ and computes $S_1 = g^{M_1} \in \mathbb{G}^{n\times n}$. Set the function index as $\mathsf{ek} = S_1$ and the associated trapdoor as $\mathsf{tk} = (g, M_1)$.
3. $\mathsf{sample}_{\mathsf{loss}}(\kappa, \mathsf{pp})$: On input pp, chooses a random matrix $M_1 \xleftarrow{\$} \mathsf{Rk}_d(\mathbb{Z}_p^{n\times n})$ and computes $S_1 = S^{M_1} \in \mathbb{G}^{n\times n}$. Set the function index as $\mathsf{ek} = S_1$.
4. $\mathsf{eval}(\mathsf{ek}, \boldsymbol{x})$: On input a function index ek and an input vector $x \in \{0,1\}^n$, compute the function $f_{\mathsf{ek}}(\boldsymbol{x}) = S_1^{\boldsymbol{x}} \in \mathbb{G}^n$.
5. $\mathsf{invert}(\mathsf{ek}, \mathsf{tk}, \boldsymbol{y})$: Given a function index $\mathsf{ek} = S_1$, the trapdoor $\mathsf{tk} = (g, M_1)$ and a vector $\boldsymbol{y} \in \mathbb{G}^n$, do the following:
   - Compute $(z_1, \cdots, z_n) = \boldsymbol{y}^{M_1^{-1}}$.
   - Let $x_i = \log_g(z_i)$ for $i = 1, \cdots, n$.
   - Output the vector $\boldsymbol{x} = (x_1, \cdots, x_n)$.

Theorem 4. *Suppose the d-Linear assumption holds for* GroupGen*. Let $p_{\max}(\kappa)$ be an upper bound on the order of the group generated by* $\mathsf{GroupGen}(\kappa)$*. Then* C-LTDF *is an $(n, (1-\epsilon)n))$-cumulative lossy trapdoor function family, provided $\epsilon > d \log_2 p_{\max}(\kappa)/n(\kappa)$.*

Proof. Firstly, it is straightforward to verify that the inversion algorithm invert correctly recovers the (unique) pre-image on injective functions.

Secondly, we proceed to show the indistinguishability of lossy and injective modes. In injective mode, we have that M_1 is statistically close to uniform, and hence the matrix $S_1 = g^{M_1}$ is also statistically close to uniform. This implies that with very high probability, the matrix S_1 is a full rank matrix. In the lossy mode, we have that $S_1 = S^{M_1}$, where both S and M_1 are rank d matrices. We first sample S uniformly at random. By the Matrix d-linear assumption (which is implied by the d-Linear assumption – see Lemma 4) it is hard for a PPT adversary to distinguish between these two ways of sampling. However, the rank of matrix S_1 is d.This is because the elements of the matrix S can be generated using the canonical generator g' (say) of $\mathbb{Z}_p$. Hence, each row of $S_1 = S^{M_1}$ can be expressed as a linear combination of the corresponding rows and columns of S and M_1 in the exponent. At this point we can again use the Matrix d-linear assumption to sample S_1 randomly. Therefore, in both modes, ek is computationally indistinguishable from uniform, and therefore the modes are indistinguishable from each other.

Thirdly, let $\mathsf{pp} = S = g^M$ and let $\mathsf{ek} = S_1 = S^{M_1}$. Let $\boldsymbol{x} \in \{0,1\}^n$. Then $\mathsf{eval}(\mathsf{ek}, \boldsymbol{x}) = S_1^{\boldsymbol{x}} = (S^{M_1})^{\boldsymbol{x}}$. Observe that, the matrix M_1 is sampled independently of $\boldsymbol{x}$. Hence, one can completely reconstruct $f_{\mathsf{ek}}(x) = S_1^{\boldsymbol{x}}$ given $(S^{\boldsymbol{x}}, M_1)$ as follows: Define $\mathsf{compress}_{\mathsf{pp}}(\boldsymbol{x}) = S^{\boldsymbol{x}}$ and $\mathsf{expand}_{\mathsf{ek}}(R) = R^{M_1}$. However, $S^{\boldsymbol{x}}$ reveals at most ϵn bits of information about x, since in the lossy mode, the image of the function $f_{\mathsf{ek}}(\cdot)$ is contained in a subgroup of $\mathbb{G}^n$ of size $p^d < 2^{\epsilon n}$. This concludes the proof of Theorem 4. □

4.2.2 Cumulative All-Lossy-But-One Trapdoor Functions. For our construction of dual mode witness maps (DMWM), we will need a richer abstraction, which we call cumulative *all-lossy-but-one* trapdoor functions (C-ALBO-TDF). These functions are associated with an additional branch space $\mathcal{B} = \{\mathcal{B}_\kappa\}_{\kappa \in \mathbb{N}}$. For a C-ALBO-TDF, almost all the branches are lossy, except for one branch which is injective. This notion of C-ALBO-TDF is actually contrary to the notion of All-But-One Lossy TDF (ABO-LTDF) defined by Peikert and Waters [46]. ABO-LTDFs are also associated with many branches, all but one of which are injective. Also, note that, we do not need any additional public parameters in the definition C-ALBO-TDF, and we require that the residual leakages of different lossy functions are "correlated" via the public key (which is shared by different functions). Now, we formally define C-ALBO-TDF and state its properties as below:

Definition 12 (C-ALBO-TDF). *Let $\kappa \in \mathbb{N}$ be the security parameter and $\ell, \alpha : \mathbb{N} \to \mathbb{N}$ be functions. Also, let $\mathcal{B} = \{\mathcal{B}_\kappa\}_{\kappa \in \mathbb{N}}$ be a collection of sets whose elements represent the branches. An* (ℓ, α)-cumulative all-lossy-but-one trapdoor function family (C-ALBO-TDF) *with branch collection $\mathcal{B}$ is given by a tuple of (probabilistic) polynomial time algorithms* $(\mathsf{sample}_{\text{c-albo}}, \mathsf{eval}_{\text{c-albo}}, \mathsf{invert}_{\text{c-albo}})$ *(the last two being deterministic), as follows:*

- **Sampling a trapdoor function with given injective branch.** *For any branch* $b^* \in \mathcal{B}$, $\mathsf{sample}_{\mathsf{c\text{-}albo}}(\kappa, b^*)$ *outputs the tuple* $(\mathsf{ek}, \mathsf{tk})$, *where* ek *is the function index and* tk *is its associated trapdoor.*
  - **(Injective branch.)** *For the branch* b^*, $\mathsf{invert}_{\mathsf{c\text{-}albo}}(\mathsf{tk}, b^*, \mathsf{eval}_{\mathsf{c\text{-}albo}}(\mathsf{ek}, b^*, x)) = x$ *for all* $x \in \{0,1\}^{\ell(\kappa)}$ *(i.e.,* $\mathsf{eval}_{\mathsf{c\text{-}albo}}(\mathsf{ek}, b^*, \cdot)$ *computes an injective function* $g_{\mathsf{ek},b^*}(\cdot)$ *over the domain* $\{0,1\}^{\ell(\kappa)}$, *and* $\mathsf{invert}_{\mathsf{c\text{-}albo}}(\mathsf{tk}, b^*, \cdot)$ *computes* $g^{-1}_{\mathsf{ek},b^*}(\cdot)$*).*
  - **(α-Cumulative Lossy branches.)** *For all* ek *there exists an (inefficient) function* $\mathsf{compress}_{\mathsf{ek}} : \{0,1\}^{\ell(\kappa)} \to R_{\mathsf{ek}}$ *with range* $|R_{\mathsf{ek}}| \leq 2^{\ell(\kappa)-\alpha(\kappa)}$, *and for all* ek, b *there exists a function* $\mathsf{expand}_{\mathsf{ek},b}(\cdot)$ *such that the following holds. For all* $b^* \in \mathcal{B}$, *all* ek *is in the support of* $\mathsf{sample}_{\mathsf{c\text{-}albo}}(\kappa, b^*)$, *all* $b \neq b^*$ *and all* $x \in \{0,1\}^{\ell(\kappa)}$, *we have*

$$\mathsf{eval}_{\mathsf{c\text{-}albo}}(\mathsf{ek}, b, x) = \mathsf{expand}_{\mathsf{ek},b}(\mathsf{compress}_{\mathsf{ek}}(x)).$$

- **Hidden injective branch.** $\forall\, b_0^*, b_1^* \in \mathcal{B}$, *the ensembles* $\{\mathsf{ek}_0 : (\mathsf{ek}_0, \mathsf{tk}_0) \leftarrow \mathsf{sample}_{\mathsf{c\text{-}albo}}(\kappa, b_0^*)\}_{\kappa\in\mathbb{N}}$ *and* $\{\mathsf{ek}_1 : (\mathsf{ek}_1, \mathsf{tk}_1) \leftarrow \mathsf{sample}_{\mathsf{c\text{-}albo}}(\kappa, b_1^*)\}_{\kappa\in\mathbb{N}}$ *are computationally indistinguishable.*

4.2.3 A Construction of C-ALBO-TDF from $i\mathcal{O}$ and C-LTDF. In this section, we present our construction of cumulative all-lossy-but-one LTDF (C-ALBO-LTDF). We show a generic transformation from C-LTDF to C-ALBO-TDF using $i\mathcal{O}$. The main idea of our construction is as follows: We obfuscate a program that has the public parameters pp of C-LTDF hardwired in it and internally it runs either $\mathsf{sample}_{\mathsf{inj}}$ or $\mathsf{sample}_{\mathsf{loss}}$ depending on the branch b. In other words, on input a branch b, it applies a pseudorandom function to b to sample a fresh lossy branch, except for the special branch b^* on which it outputs a hard-coded injective C-LTDF key. The details of our construction follows.

The construction. Let $\text{C-LTDF} = (\mathsf{setup}, \mathsf{sample}_{\mathsf{inj}}, \mathsf{sample}_{\mathsf{loss}}, \mathsf{eval}, \mathsf{invert})$ be collection of (n, α)-cumulative lossy trapdoor function (C-LTDF) and let $i\mathcal{O}$ be indistinguishability obfuscator for all circuits. Also, let $\mathcal{R}_{\mathsf{inj}}$ and $\mathcal{R}_{\mathsf{loss}}$ be the randomness spaces of the algorithms $\mathsf{sample}_{\mathsf{inj}}$ and $\mathsf{sample}_{\mathsf{loss}}$ respectively. We now present our construction (see Figure 3).

1. $\mathsf{sample}_{\mathsf{c\text{-}albo}}(\kappa, b^*)$ outputs the function index ek' which contains the obfuscation of a program $\mathsf{Samp\text{-}index}$ (described below). It also outputs an associated trapdoor tk'. The output of the obfuscated program is a function index.
2. $\mathsf{eval}_{\mathsf{c\text{-}albo}}(\mathsf{ek}', b, x)$ runs the obfuscated program in ek' on input b to obtain a function index $\widetilde{\mathsf{ek}}$ and outputs $y = \mathsf{eval}(\widetilde{\mathsf{ek}}, x)$.
3. $\mathsf{invert}_{\mathsf{c\text{-}albo}}(\mathsf{tk}', b, y)$ can be used to efficiently invert y if $b = b^*$.

The program $\mathsf{Samp\text{-}index}$ has a PRF key K, the injective branch b^*, ek and pp hard-coded in it. On input b, if $b = b^*$ it outputs the injective function ek. Else, it outputs $\mathsf{sample}_{\mathsf{loss}}(\kappa, \mathsf{pp}; r_{\mathsf{loss}})$, where $r_{\mathsf{loss}} \leftarrow \mathsf{PRF}(K, b)$. The function index ek' is set to be the obfuscation of the above program $\mathsf{Samp\text{-}index}$, and the trapdoor is set to be tk, which is sampled as part of $\mathsf{sample}_{\mathsf{c\text{-}albo}}$.

Let C-LTDF $= (\mathsf{setup}, \mathsf{sample}_{\mathsf{inj}}, \mathsf{sample}_{\mathsf{loss}}, \mathsf{eval}, \mathsf{invert})$ be collection of (ℓ, α)-C-LTDF with randomness paces $\mathcal{R}_{\mathsf{inj}}$ and $\mathcal{R}_{\mathsf{loss}}$ for the algorithms $\mathsf{sample}_{\mathsf{inj}}$ and $\mathsf{sample}_{\mathsf{loss}}$ respectively. Let $F : \mathcal{K} \times \mathcal{B} \to \mathcal{R}_{\mathsf{loss}}$ be a puncturable PRF given by a pair of polynomial time algorithms $(F.\mathsf{puncture}, F.\mathsf{eval})$ and $i\mathcal{O}$ be an indistinguishability obfuscator for circuits.

1. $\mathsf{sample}_{\mathsf{c\text{-}albo}}(\kappa, b^*)$: Run $\mathsf{setup}(\kappa)$ to obtain the public parameters pp. Then it samples $(\mathsf{ek}, \mathsf{tk}) \leftarrow \mathsf{sample}_{\mathsf{inj}}(\kappa, \mathsf{pp})$. Then create an obfuscated program $R \leftarrow i\mathcal{O}(\mathsf{Samp\text{-}index}_{K,b^*,\mathsf{ek},\mathsf{pp}})$, where the program $\mathsf{Samp\text{-}index}_{K,b^*,\mathsf{ek},\mathsf{pp}}$ is as shown below. Output $\mathsf{ek}' = R$ and $\mathsf{tk}' = \mathsf{tk}$.
2. $\mathsf{eval}_{\mathsf{c\text{-}albo}}(\mathsf{ek}', b, x)$: Here $x \in \{0,1\}^n$. Parse ek' as R. Run $\widetilde{\mathsf{ek}} \leftarrow R(b)$, and output the value $y = \mathsf{eval}(\widetilde{\mathsf{ek}}, x)$.
3. $\mathsf{invert}_{\mathsf{c\text{-}albo}}(\mathsf{tk}', b^*, y)$: Parse tk' as tk, and compute $\mathsf{invert}(\mathsf{tk}, y)$.

Program $\mathsf{Samp\text{-}index}_{K,b^*,\mathsf{ek},\mathsf{pp}}(b)$

Constant: PRF key K, branch b^*, (injective) function index ek and public parameters pp.

Input Domain: $b \in \mathcal{B}$

if $b = b^*$ **then**

output $\widetilde{\mathsf{ek}} := \mathsf{ek}$.

else

compute $r_{\mathsf{loss}} \leftarrow F(K, b)$. Sample $(\widetilde{\mathsf{ek}}, \perp) \leftarrow \mathsf{sample}_{\mathsf{loss}}(\kappa, \mathsf{pp}; r_{\mathsf{loss}})$,

output $\widetilde{\mathsf{ek}}$.

Fig. 3. Construction of C-ALBO-TDF. The program $\mathsf{Samp\text{-}index}_{K,b^*,\mathsf{ek},\mathsf{pp}}$ is padded to the the maximum of the size of itself and the programs $\mathsf{Samp\text{-}index}_{K,b^*,\mathsf{ek}^*,\mathsf{pp}}$ (defined in Figure 4) and $\mathsf{Samp\text{-}index}_{\widehat{K}_{b^*},b^*,\mathsf{ek}^*,\mathsf{pp}}$ (defined in Figure 5).

Theorem 5. *Let* C-LTDF *be a collection of* (ℓ, α)*-cumulative LTDF,* $i\mathcal{O}$ *be an indistinguishability obfuscator for circuits,* F *be a secure puncturable PRF with input space* $\mathcal{B}$*. Then the construction* C-ALBO-TDF *defined in* Figure 3 *is a collection of* (ℓ, α)*-cumulative all-lossy-but-one trapdoor functions.*

Proof. The correctness of the inversion algorithm $\mathsf{invert}_{\mathsf{c\text{-}albo}}$ follows from the correctness of eval when evaluated on the injective function index.

Lemma 5. *The above construction* C-ALBO-TDF *achieves hidden injective branch property.*

Proof. To prove the hidden injective branch property of C-ALBO-TDF, we consider the following hybrid experiments described below. Also, let us denote by S_i the probability that $\mathcal{A}$ wins in Hybrid i.

Hybrid 0. This corresponds to the original security game. In particular, the challenger runs $\mathsf{setup}(\kappa)$ to obtain the public parameters pp. Then it samples $(\mathsf{ek}, \mathsf{tk}) \leftarrow \mathsf{sample}_{\mathsf{inj}}(\kappa, \mathsf{pp})$ and creates an obfuscated program $R \leftarrow i\mathcal{O}(\mathsf{Samp\text{-}index}_{K,b^*,\mathsf{ek},\mathsf{pp}})$, as in the original construction. It then outputs the function index $\mathsf{ek}' = R$.

Hybrid 1. In this hybrid, we change the way the function index ek' is derived. In particular, we derive ek' using a modified $\mathsf{sample}_{\mathsf{c\text{-}albo}}$: In this modified $\mathsf{sample}_{\mathsf{c\text{-}albo}}$, the

challenger only runs the lossy sampler irrespective of the branch b used as input to the algorithm $\mathsf{eval}_{\mathsf{c\text{-}albo}}$. In particular, the challenger does the following: On input (κ, b^*), computes $r^* \leftarrow F(K, b^*)$ and use r^* to sample ek^* as $(\mathsf{ek}^*, \perp) \leftarrow \mathsf{sample}_{\mathsf{loss}}(\kappa, \mathsf{pp}; r^*)$. Then it creates an obfuscated program $R_1 \leftarrow i\mathcal{O}(\mathsf{Samp\text{-}index}_{K,b^*,\mathsf{ek}^*,\mathsf{pp}})$, as defined in Figure 4. Output $\mathsf{ek}' = R_1$.

Program $\mathsf{Samp\text{-}index}_{K,b^*,\mathsf{ek}^*,\mathsf{pp}}(b)$

Constant: PRF key K, branch b^*, ek^* and pp.

Input Domain: $b \in \mathcal{B}$

if $b = b^*$ **then**

output ek^*.

else

compute $r_{\mathsf{loss}} \leftarrow F(K, b)$. Sample $(\widetilde{\mathsf{ek}}, \perp) \leftarrow \mathsf{sample}_{\mathsf{loss}}(\kappa, \mathsf{pp}; r_{\mathsf{loss}})$,

output $\widetilde{\mathsf{ek}}$.

Fig. 4. The program $\mathsf{Samp\text{-}index}_{K,b^*,\mathsf{ek}^*,\mathsf{pp}}$

Claim. $|\Pr[S_1] - \Pr[S_0]| \leq \mathsf{negl}(\kappa)$.

Proof. The proof of this claim follows from the indistinguishability of injective and lossy branches of C-LTDF. In particular, if the adversary $\mathcal{A}$ has a non-negligible advantage in distinguishing between Hybrid 0 and Hybrid 1, we can construct an adversary $\mathcal{B}$ that breaks the indistinguishability of the injective and lossy branches property of C-LTDF. The adversary $\mathcal{B}$ receives as input $(\mathsf{pp}, \widetilde{\mathsf{ek}})$ from its challenger and constructs the obfuscated program R_1 (by sampling a PRF key K by itself). Finally, it outputs the program R_1 as the function index ek' to $\mathcal{A}$. If the function index $\widetilde{\mathsf{ek}}$ received by $\mathcal{B}$ was injective, this corresponds to Hybrid 0; else if ek was lossy, it corresponds to Hybrid 1. Hence, if the advantage of $\mathcal{A}$ is non-negligible, so is the advantage of $\mathcal{B}$, which contradicts the indistinguishability property of injective and lossy branches of C-LTDF (even given the public parameters pp). □

Hybrid 2: This is similar to Hybrid 1, except that we further change the way the function index ek' is derived: On input (κ, b^*), the challenger computes $r^* \leftarrow F(K, b^*)$ and samples ek^* as $(\mathsf{ek}^*, \perp) \leftarrow \mathsf{sample}_{\mathsf{loss}}(\kappa, \mathsf{pp}; r^*)$ as before. Next, it $\widehat{K}_{b^*} \leftarrow F.\mathsf{puncture}(K, b^*)$, and creates the obfuscated $R_2 \leftarrow i\mathcal{O}(\mathsf{Samp\text{-}index}_{\widehat{K}_{b^*},b^*,\mathsf{ek}^*,\mathsf{pp}})$, where the program $\mathsf{Samp\text{-}index}_{\widehat{K}_{b^*},b^*,\mathsf{ek}^*,\mathsf{pp}}$ is defined in Figure 5.

Claim. $|\Pr[S_2] - \Pr[S_1]| \leq \mathsf{negl}(\kappa)$.

Proof. The only difference between Hybrid 1 and Hybrid 2 is that: in Hybrid 2 the punctured key $\widehat{K}_{b^*}$ is used, instead of using the master PRF key K. However, the correctness of punctured PRF guarantees that on branches $b \neq b^*$, $F.\mathsf{eval}(\widehat{K}_{b^*}, b) = F(K, b)$. Hence,

Program $\mathsf{Samp\text{-}index}_{\widehat{K}_{b^*},b^*,\mathsf{ek}^*,\mathsf{pp}}(b)$

Constant: Punctured PRF key $\widehat{K}_{b^*}$, branch b^*, ek^*, pp.

Input Domain: $b \in \mathcal{B}$

if $b = b^*$ **then**

output ek^*.

else

compute $r_{\mathsf{loss}} \leftarrow F.\mathsf{eval}(\widehat{K}_{b^*}, b)$. Sample $(\widetilde{\mathsf{ek}}, \perp) \leftarrow \mathsf{sample}_{\mathsf{loss}}(\kappa, \mathsf{pp}; r_{\mathsf{loss}})$,

output $\widetilde{\mathsf{ek}}$.

Fig. 5. The program $\mathsf{Samp\text{-}index}_{\widehat{K}_{b^*},b^*,\mathsf{ek}^*,\mathsf{pp}}$.

both the programs behave identically. Hence, the security of $i\mathcal{O}$ guarantees that both these hybrids are computationally indistinguishable. □

Hybrid 3: This is similar to Hybrid 2, except that the challenger now samples r^* uniformly at random from $\mathcal{R}_{\mathsf{loss}}$, instead of computing it using $F(K, b^*)$, and uses this r^* to sample ek^* (using $\mathsf{sample}_{\mathsf{loss}}$ as before). The rest of the computations remain the same as in Hybrid 2.

Claim. $|\Pr[S_3] - \Pr[S_2]| \leq \mathsf{negl}(\kappa)$.

Proof. The claim follows in a straightforward way from the pseudorandomness of the puncturable PRF F. □

Claim. $\Pr[S_3] = \frac{1}{2}$.

Proof. Note that, in Hybrid 3, the branch information (b_0^* or b_1^*) is not used by $\mathsf{sample}_{\mathsf{loss}}$. Hence, the claim follows. □

Lemma 6. *The above construction* C-ALBO-TDF *is* (ℓ, α)*-cumulative lossy.*

Proof. This follows from the (ℓ, α)-cumulative lossiness of C-LTDF. Note that, on a branch $b \neq b^*$ the algorithm $\mathsf{sample}_{\mathsf{c\text{-}albo}}(\mathsf{ek}, b, x)$ just runs the algorithm $\mathsf{eval}(\widetilde{\mathsf{ek}}, x)$, where $\widetilde{\mathsf{ek}}$ is a lossy function index derived by running the obfuscated program R on input b.

Combining Lemma 5 and Lemma 6, we get the complete proof of Theorem 5. □

4.3 Construction of Dual Mode Witness Maps

In this section, we present a construction of dual mode witness maps (DMWM) for any **NP** relation R_ℓ (see Figure 6). The main building blocks of our construction are an appropriately lossy compact witness map (CWM) and a cumulatively all-lossy-but-one trapdoor function (C-ALBO-TDF).

Intuition behind the construction. The CRS of DMWM will consist of the function index ek of C-ALBO-TDF sampled using the special injective tag tag^* (we require that

the tag space of DMWM is same as the branch space of C-ALBO-TDF) as wells as a CRS of CWM. To compute a proof for a statement x with witness w under a tag tag, the prover computes $Y = \mathsf{eval}_{\text{c-albo}}(\mathsf{ek}, \mathsf{tag}, w)$ and then uses the CWM to prove that Y was computed correctly using a valid witness w for the statement x. The completeness and soundness of DMWM follows directly from the completeness and soundness guarantees of CWM. The cumulative lossiness of DMWM follows from the cumulative lossiness of CWM and C-ALBO-TDF.

(a) Let C-ALBO-TDF $= (\mathsf{sample}_{\text{c-albo}}, \mathsf{eval}_{\text{c-albo}}, \mathsf{invert}_{\text{c-albo}})$ be collection of $(\ell, (\ell - \alpha'))$-C-ALBO-TDF, with branch space $\mathcal{B}$.
(b) Let CWM $= (\text{CWM.}\mathsf{setup}, \text{CWM.}\mathsf{map}, \text{CWM.}\mathsf{check})$ be a α-CWM (please refer to Section 4) for the following language:

$$L := \left\{ (x, \mathsf{ek}, \mathsf{tag}, Y) \ : \ \exists\ w \ \text{s.t.} \ \left(Y = \mathsf{eval}_{\text{c-albo}}(\mathsf{ek}, \mathsf{tag}, w) \ \wedge \ (x, w) \in R_\ell \right) \right\}$$

We construct DMWM $= (\text{DMWM.}\mathsf{setup}, \text{DMWM.}\mathsf{map}, \text{DMWM.}\mathsf{check}, \text{DMWM.}\mathsf{extract})$ with tag space $\mathcal{T} = \mathcal{B}$ for the **NP** relation R_ℓ as follows:

1. $\text{DMWM.}\mathsf{setup}(\kappa, \ell, \mathsf{tag})$: Here $\mathsf{tag} \in \mathcal{T}$. Run $\text{CWM.}\mathsf{setup}(\kappa, \ell)$ to output a string K' of length polynomial in the security parameter κ. Also, run $\mathsf{sample}_{\text{c-albo}}(\kappa, \mathsf{tag})$ to output the tuple $(\mathsf{ek}, \mathsf{tk})$. Set $\mathsf{K} = (\mathsf{K}', \mathsf{ek})$ and the trapdoor $\mathsf{td} = \mathsf{tk}$.
2. $\text{DMWM.}\mathsf{map}(\mathsf{K}, \mathsf{tag}', x, w)$: Here $\mathsf{tag}' \in \mathcal{T}$. Parse K as $\mathsf{K} = (\mathsf{K}', \mathsf{ek})$, and do the following:
   - Compute $Y = \mathsf{eval}_{\text{c-albo}}(\mathsf{ek}, \mathsf{tag}', w)$, and
   - Compute $w^*_{\text{CWM}} = \text{CWM.}\mathsf{map}(\mathsf{K}', (x, \mathsf{ek}, \mathsf{tag}', Y), w)$.

   Output the representative witness $w^* = (Y, w^*_{\text{CWM}})$.
3. $\text{DMWM.}\mathsf{check}(\mathsf{K}, \mathsf{tag}', x, w^*)$: Parse $\mathsf{K} = (\mathsf{K}', \mathsf{ek})$ and $w^* = (Y, w^*_{\text{CWM}})$. Output $\text{CWM.}\mathsf{check}(\mathsf{K}', (x, \mathsf{ek}, \mathsf{tag}', Y), w^*_{\text{CWM}})$.
4. $\text{DMWM.}\mathsf{extract}(\mathsf{td}, x, w^*)$: Parse $w^* = (Y, w^*_{\text{CWM}})$. Output $\mathsf{invert}_{\text{c-albo}}(\mathsf{td}, \mathsf{tag}, Y)$, where $(\mathsf{ek}, \mathsf{tk}) \leftarrow \mathsf{sample}_{\text{c-albo}}(\kappa, \mathsf{tag})$ was generated as part of setup using the same tag tag.

Fig. 6. Construction of DMWM for an **NP** relation R_ℓ.

Theorem 6. *Let $\alpha, \alpha' \geq 0$, and $\alpha'' = (\alpha + \alpha')$. Let* CWM *be a (selectively) sound α-CWM for the* **NP** *language L,* C-ALBO-TDF *let a collection of $(\ell, (\ell - \alpha'))$-cumulative all-lossy-but-one LTDF with branch space $\mathcal{B}$. Then the construction* DMWM *defined in* Figure 6 *is α''-DMWM with tag space $\mathcal{T} = \mathcal{B}$ for the* **NP** *relation R_ℓ.*

Proof sketch: Before providing the proof, we provide a proof sketch here. The completeness condition of DMWM follows in a straightforward manner from the correctness of the underlying CWM. The (selective) soundness of DMWM follows from the (selective) soundness of CWM. The hidden tag property of DMWM follows from the hidden injective branch property of the underlying C-ALBO-TDF. Finally, the α''-lossiness of DMWM follows from the cumulative loss parameters of α-CWM and $(\ell, (\ell - \alpha'))$-C-ALBO-TDF. □

Proof. We show that the construction of DMWM shown in Figure 6 is a α''-DMWM for R_ℓ, when CWM is a α-CWM and C-ALBO-TDF is a $(\ell, (\ell - \alpha'))$-ALBO-TDF. To show this, we need to prove the following properties:

1. *Extraction:* We show that if there is a PPT adversary $\mathcal{A}$ such that the advantage $\mathsf{Adv}^{\text{DMWM}}_{\mathcal{A}}(\kappa)$ is non-negligible, then there is a PPT adversary $\mathcal{A}^*$ such that $\mathsf{Adv}^{\text{CWM}}_{\mathcal{A}^*}(\kappa)$ is also non-negligigble.
   $\mathcal{A}^*$ internally runs $\mathcal{A}$ and externally interacts with a challenger as in the definition of $\mathsf{Adv}^{\text{CWM}}_{\mathcal{A}^*}(\kappa)$, as follows:
   - When $\mathcal{A}$ responds with tag, $\mathcal{A}^*$ internally runs $\mathsf{sample}_{\text{c-albo}}(\kappa, \mathsf{tag})$ to obtain $(\mathsf{ek}, \mathsf{tk})$. Then it sets $\mathsf{K} = (\mathsf{K}', \mathsf{ek})$, and passes it to $\mathcal{A}$.
   - When $\mathcal{A}$ responds with (x^*, w^*), $\mathcal{A}^*$ parses w^* as (Y, w^*_{CWM}), and outputs $((x^*, \mathsf{ek}, \mathsf{tag}, Y), w^*_{\text{CWM}})$ to the challenger.

   We shall show that $\mathsf{Adv}^{\text{CWM}}_{\mathcal{A}^*}(\kappa) \geq \mathsf{Adv}^{\text{DMWM}}_{\mathcal{A}}(\kappa)$. Let $\hat{w} = \text{DMWM.}\mathsf{extract}(\mathsf{td}, x^*, w^*)$, where $\mathsf{td} = \mathsf{tk}$ was generated along with ek, by $\mathcal{A}^*$. By the definition of $\mathsf{Adv}^{\text{DMWM}}_{\mathcal{A}}(\kappa)$, it would be the case that $\text{DMWM.}\mathsf{check}(\mathsf{K}, \mathsf{tag}, x^*, w^*) = 1$ and $(x^*, \hat{w}) \notin R$ with probability $\mathsf{Adv}^{\text{DMWM}}_{\mathcal{A}}(\kappa)$. That is, with that probability, $w^* = (Y, w^*_{\text{CWM}})$, where $\text{CWM.}\mathsf{check}(\mathsf{K}', (x^*, \mathsf{ek}, \mathsf{tag}, Y), w^*_{\text{CWM}}) = 1$, but $(x^*, \hat{w}) \notin R$. Now, by definition of $\text{DMWM.}\mathsf{extract}$, we have $\hat{w} = \mathsf{invert}_{\text{c-albo}}(\mathsf{tk}, \mathsf{tag}, Y)$. Since tag corresponds to the injective branch of C-ALBO-TDF, we are guaranteed that $\mathsf{invert}_{\text{c-albo}}(\mathsf{tk}, \mathsf{tag}, Y)$ is the unique value $\hat{w}$ such that $\mathsf{eval}_{\text{c-albo}}(\mathsf{ek}, \mathsf{tag}, \hat{w}) = Y$. Hence if $(x^*, \hat{w}) \notin R$, then $(x^*, \mathsf{ek}, \mathsf{tag}, Y) \notin L$.
   Thus, with probability $\mathsf{Adv}^{\text{DMWM}}_{\mathcal{A}}(\kappa)$, the adversary $\mathcal{A}^*$ outputs a pair $((x^*, \mathsf{ek}, \mathsf{tag}, Y), w^*_{\text{CWM}})$ such that

   $$(x^*, \mathsf{ek}, \mathsf{tag}, Y) \notin L, \qquad \text{CWM.}\mathsf{check}(\mathsf{K}', (x^*, \mathsf{ek}, \mathsf{tag}, Y), w^*_{\text{CWM}}) = 1.$$

   In other words, $\mathsf{Adv}^{\text{CWM}}_{\mathcal{A}^*}(\kappa) \geq \mathsf{Adv}^{\text{DMWM}}_{\mathcal{A}}(\kappa)$, as required to show.

2. *Hidden Tag:* Consider a PPT adversary $\mathcal{A}_{\text{DMWM-hide}}$ in the definition of $\mathsf{Adv}^{\text{DMWM-hide}}_{\mathcal{A}}(\kappa)$. We shall describe another adversary $\mathcal{A}_{\text{CWM-hide}}$, which internally runs $\mathcal{A}_{\text{DMWM-hide}}$ and successfully breaks the hidden injective branch property of C-ALBO-TDF with the same advantage as $\mathcal{A}_{\text{DMWM-hide}}$. In particular, $\mathcal{A}_{\text{CWM-hide}}$ does the following: On input two distinct tags tag_0 and tag_1 from $\mathcal{A}_{\text{DMWM-hide}}$, $\mathcal{A}_{\text{CWM-hide}}$ forwards both tag_0 and tag_1 to its challenger. It then receives ek_b corresponding to one of these tags. It then runs $\mathsf{K}' \leftarrow \text{CWM.}\mathsf{setup}(\kappa, \ell)$, and returns the tuple $\mathsf{K} = (\mathsf{K}', \mathsf{ek}_b)$ to $\mathcal{A}_{\text{DMWM-hide}}$. At some point, $\mathcal{A}_{\text{DMWM-hide}}$ outputs a bit b' as a guess for which of the tags was used. The adversary $\mathcal{A}_{\text{CWM-hide}}$ also outputs the same bit b'. If the advantage of $\mathcal{A}_{\text{DMWM-hide}}$ non-negligible, then so is the advantage of $\mathcal{A}_{\text{CWM-hide}}$.

3. *α''-lossiness:* Note that, in our construction we assume that C-ALBO-TDF is a collection of $(\ell, (\ell - \alpha'))$-C-ALBO-TDF. This implies that for all $\mathsf{tag}' \neq \mathsf{tag}$ (where tag is used in the algorithm $\mathsf{sample}_{\text{c-albo}}$ for sampling the function index ek) $\mathsf{eval}_{\text{c-albo}}(\mathsf{ek}, \mathsf{tag}', w) = \mathsf{expand}_{\mathsf{ek},\mathsf{tag}'}(\mathsf{compress}_{\mathsf{ek}}(w))$, where $\mathsf{compress}_{\mathsf{ek}} : \{0,1\}^{\ell(\kappa)} \to R_{\mathsf{ek}}$ and $|R_{\mathsf{ek}}| \leq 2^{\alpha'(\kappa)}$.
   Also, in our construction the CWM CWM is α-lossy. This implies that, $\forall \mathsf{K}' \leftarrow \text{CWM.}\mathsf{setup}(\kappa, \ell)$, $\forall x \in \{0,1\}^\ell$, $|\{\text{CWM.}\mathsf{map}(\mathsf{K}', x, w) \mid (x, w) \in R_\ell\}| \leq 2^{\alpha(\kappa)}$. Expressed in terms of the

functions $\mathsf{compress}_{\mathsf{K}',x}(\cdot)$ and $\mathsf{expand}_{\mathsf{K}',x}(\cdot)$, we get that $\mathsf{compress}_{\mathsf{K}',x}(\cdot) = \text{CWM.}\mathsf{map}(\mathsf{K}', x, \cdot)$ and $\mathsf{expand}_{\mathsf{K}',x}(\cdot) = \mathsf{id}(\cdot)$, where $\mathsf{id}(\cdot)$ is the identity function. Here, $\mathsf{compress}_{\mathsf{K}',x} : \{0,1\}^* \to R_{\mathsf{K}',x}$ where $|R_{\mathsf{K}',x}| \leq 2^{\alpha(\kappa)}$.
The representative witnesses for DMWM w^* consist of a tuple of values produced by both CWM and C-ALBO-TDF. Hence, for DMWM, we have that: $\mathsf{compress}_{\mathsf{K},x} : \{0,1\}^* \to S_{\mathsf{K},x}$, where the function $\mathsf{compress}_{\mathsf{K},x}(\cdot) = \big(\mathsf{compress}_{\mathsf{ek}}(\cdot), \mathsf{compress}_{\mathsf{K}',x}(\cdot)\big)$, and $S_{\mathsf{K},x} = R_{\mathsf{ek}} \cup R_{\mathsf{K}',x}$. Moreover the function $\mathsf{expand}_{\mathsf{K},x}(\cdot) = \big(\mathsf{expand}_{\mathsf{ek},\mathsf{tag}'}(\cdot), \mathsf{expand}_{\mathsf{K}',x}(\cdot)\big)$. Hence, we have that $|S_{\mathsf{K},x}| \leq 2^{\alpha+\alpha'} = 2^{\alpha''}$.

This concludes the proof of the above theorem. □

5 Fully Leakage and Tamper-resilient Signature Scheme

A signature scheme with setup SIG is a tuple of PPT algorithms SIG = ($\mathsf{setup}, \mathsf{keygen}, \mathsf{sign}, \mathsf{verify}$). The setup algorithm takes as input the security parameter κ, and outputs a set of public parameters pub, which is taken as an implicit input (along with κ) by all the other algorithms. We denote the message space (implicitly parametrized by κ) as $\mathcal{M}$. We shall require *perfect correctness*: For all $\mathsf{pub} \leftarrow \text{SIG.}\mathsf{setup}(\kappa)$, any key pair $(\mathsf{ssk}, \mathsf{vk})$ produced by SIG.keygen and all messages $m \in \mathcal{M}$, we require $\text{SIG.}\mathsf{verify}(\mathsf{vk}, (m, \text{SIG.}\mathsf{sign}(\mathsf{ssk}, m))) = 1$.

We define fully-leakage and tamper-resilient (FLTR) signature security, in the bounded leakage and tampering model. Before defining the model formally, we provide an informal description here. In this model, first the challenger sets up the public parameters pub, and also generates a key-pair $(\mathsf{ssk}, \mathsf{vk})$. Then, vk is given to the adversary, and as in the case of standard signature security experiment, the adversary is given access to a signing oracle and it attempts to produce a valid signature on a message which it has not queried. But in addition, the adversary has access to a leakage oracle and a tampering oracle, as described below. Leakage and tampering act on st, which consists of the signing key ssk and all the randomness used by the signing algorithm thus far. Note that here, for definitional purposes, we allow SIG.sign to be randomized, though in our construction it will be deterministic.

Leakage: The adversary can adaptively query the leakage oracle with any efficiently computable functions f and will receive $f(\mathsf{st})$ in return (subject to bounds below).

Tampering: The adversary can adaptively query the tampering oracle with efficiently computable functions T, and on each such query, the tampering oracle will generate a signing key and randomness for signature: $(\widetilde{\mathsf{ssk}}, \widetilde{r}) = T(\mathsf{st})$. Subsequently, the adversary can adaptively query each signing oracle $\text{SIG.}\mathsf{sign}(\widetilde{\mathsf{ssk}}, \cdot, \widetilde{r})$, any number of times (subject to bounds below).

Bounds on Queries: The total output length of all the leakage functions ever queried to the leakage oracle is bounded by $\lambda(\kappa)$. For tampering, there is an upper bound $t(\kappa)$ on the total number of tampering functions queried by the adversary. However, the adversary may ask an unbounded number of untampered or tampered signing queries to the signing oracle. We shall denote an FLTR signature scheme with security subject to these bounds as (λ, t)-FLTR signature scheme.

5.1 Security model for FLTR signatures.

Definition 13. ((λ, t)-FLTR security). *We say that a signature scheme* $\text{SIG} = (\text{SIG.}\mathsf{setup}, \text{SIG.}\mathsf{keygen}, \text{SIG.}\mathsf{sign}, \text{SIG.}\mathsf{verify})$ *is* (λ, t)-fully-leakage and tamper-resilient (FLTR) *if for all PPT adversaries/forgers* $\mathcal{F}$ *there exists a negligible function* $\mathsf{negl} : \mathbb{N} \to \{0,1\}$ *such that* $\Pr\left[\mathsf{Success}^{(\lambda,t)\text{-}\mathsf{FLTR}}_{\Pi,\mathcal{F}}(\kappa)\right] \leq \mathsf{negl}(\kappa)$*, where the event* $\mathsf{Success}^{(\lambda,t)\text{-}\mathsf{FLTR}}_{\Pi,\mathcal{F}}(\kappa)$ *is defined via the following experiment between a challenger* $\mathcal{C}$ *and the forger* $\mathcal{F}$*:*

1. *Initially, the challenger* $\mathcal{C}$ *computes* $\mathsf{pub} \leftarrow \text{SIG.}\mathsf{setup}(\kappa)$ *and* $(\mathsf{ssk}, \mathsf{vk}) \leftarrow \text{SIG.}\mathsf{keygen}(\kappa, \mathsf{pub})$*, and sets* $\mathsf{st} = \mathsf{ssk}$*.*
2. *The forger on receiving* pub *and* vk*, can adaptively query the following oracles as defined below:*
   - **Signing queries**: *The signing oracle* $\text{SIG.}\mathsf{sign}^*_{\mathsf{ssk}}(\cdot)$ *receives as input a message* $m_i \in \mathcal{M}$*. The challenger* $\mathcal{C}$ *then samples* $r_i \leftarrow \mathcal{R}$*, and computes* $\sigma_i \leftarrow \text{SIG.}\mathsf{sign}(\mathsf{ssk}, m_i, r_i)$*. It appends* r_i *to* st *and outputs* σ_i*.*
   - **Leakage queries**: *The leakage oracle receives as input (the description of) an efficiently computable function* $f_j : \{0,1\}^* \to \{0,1\}^{\lambda_j}$*, and responds with* $f_j(\mathsf{st})$*.*
   - **Tampering queries**: *When the forger* $\mathcal{F}$ *(adaptively) submits the* i^{th} *tampering query* T_i*, the challenger computes* $(\widetilde{\mathsf{ssk}}_i, \widetilde{r}_i) = T_i(\mathsf{st})$*. Subsequently,* $\mathcal{F}$ *can adaptively query the tampered-signing oracle* $\text{SIG.}\mathsf{sign}(\widetilde{\mathsf{ssk}}_i, \cdot, \widetilde{r}_i)$ *using messages in* $\mathcal{M}$*. We call these as "tampered signing queries".*
3. *Eventually,* $\mathcal{F}$ *outputs a message-signature pair* (m^*, σ^*) *as the purported forgery.*

$\mathsf{Success}^{(\lambda,t)\text{-}\mathsf{FLTR}}_{\Pi,\mathcal{F}}(\kappa)$ *denotes the event in which the following happens:*

- *The signature* σ^* *verifies with respect to the original verification key* vk*, i.e.,* $\text{SIG.}\mathsf{verify}(\mathsf{vk}, (m^*, \sigma^*)) = 1$*.*
- m^* *was never queried as input to the signing or tampered signing oracle by the forger* $\mathcal{F}$*.*
- *The output length of all the leakage functions* $\sum_j \lambda_j$ *is at most* $\lambda(\kappa)$*.*
- *The number of tampering queries made by* $\mathcal{F}$ *is at most* $t(\kappa)$*.*

We also consider a selective variant of the above definition, where the message m^* (on which the forgery is to be produced) is declared by the adversary before receiving the public parameters pub and the verification key vk. We call this *selectively unforgeable* (λ, t)-FLTR signature scheme. We shall focus on this model in our construction (see Section 5.2) and note that one can convert a selectively unforgeable (λ, t)-FLTR signature scheme to an adaptively secure one by relying on complexity leveraging, when appropriate.

Remark 2. Note that our definition of FLTR signatures is very general and encompasses all other previous definitions. When $(\lambda, t) = (0, 0)$, we obtain the original notion of existential unforgeability under adaptive chosen message attacks (EUF-CMA). When

$(\lambda, t) = (\lambda, 0)$, we recover the definition of fully leakage-resilient (FLR) signature definition proposed by Boyle, Segev and Wichs [11]. When the adversary can leak from or tamper with *only* the secret signing key (and *not* the randomness used by the signer), i.e., $\mathsf{st} = \{\mathsf{ssk}\}$ only, we obtain the definition of leakage and tamper-resilient signature schemes, as defined by Damgård et al. [15] and by Faonio and Venturi [21].

We remark that the above definition can be slightly simplified if we require that the signature scheme is deterministic. While our construction is indeed deterministic, we present the above definition that applies to randomized schemes as well, to explicate what *full* leakage and tamper resilience entails.

Remark 3. If the construction of the FLTR signature scheme is *deterministic* (in which case we simply call it a deterministic LTR signature scheme), proving security for such a scheme in our new security scheme essentially reduces to proving security in the model of [21], since there is no randomness to leak from or tamper with. However, we stress that for a randomized signature scheme our model is *strictly stronger* than the model of [21] or [11].

5.2 Construction of our FLTR signature scheme.

In this section, we present our construction of FLTR signature scheme. In Figure 7, we present this construction.

Theorem 7. *Let $\lambda(\kappa)$, $t(\kappa)$, $d(\kappa)$ and $m(\kappa)$ be parameters. Let* SPR *be a second pre-image resistant function mapping $d(\kappa)$ bits to $m(\kappa)$ bits, and* DMWM *be a κ-lossy* DMWM *with tag space $\mathcal{T} = \mathcal{M}$ (where $\mathcal{M}$ is the message space of* SIG*). Then the above construction* SIG *is a $\big(\lambda(\kappa), t(\kappa)\big)$-*FLTR *signature scheme, as long as the parameters satisfy:*

$$0 \leq \lambda(\kappa) \leq d(\kappa) - \kappa\big(t(\kappa)+1\big) - m(\kappa) - \omega(\log \kappa).$$

Hence, the relative leakage rate is $\frac{\lambda(\kappa)}{d(\kappa)} \approx 1 - \frac{\kappa(t(\kappa)+1)-m(\kappa)-\omega(\log\kappa)}{d(\kappa)} = 1 - o(1)$, for an appropriate choice of $\big(\kappa(t(\kappa)+1) - m(\kappa) - \omega(\log\kappa)\big) = o(d(\kappa))$. The tampering rate $\rho(\kappa)$ is $\rho(\kappa) = \frac{t(\kappa)}{d(\kappa)} = O(1/\kappa)$.

Proof sketch: Before presenting the detailed proof, we present a high level idea behind our proof strategy here. Suppose there is an adversary who can break the security of the signature scheme SIG. We use this adversary to break the security of the SPR function SPR. The signing key of our LTR construction is a random pre-image x of SPR, and the public parameters pub are sampled honestly as in the construction. In the *first* hybrid, we change the way the public parameters pub are generated: In particular, we run $\text{DMWM}.\mathsf{setup}(\kappa, \ell, m^*)$ on the tag m^* (recall that in the selective unforgeability game, m^* is declared before the setup phase) to derive the string K of DMWM. Additionally, instead of discarding the trapdoor td while sampling K, the challenger now retains td along with the string K while running the setup algorithm $\text{DMWM}.\mathsf{setup}(\kappa, \ell, m^*)$. The *hidden tag* property of DMWM ensures that this hybrid is computationally indistinguishable from the

1. Let $\text{SPR} = (\text{SPR.}\mathsf{gen}, \text{SPR.}\mathsf{eval})$ be a family of SPR functions from $\{0,1\}^{d(\kappa)}$ to $\{0,1\}^{m(\kappa)}$, where $m(\kappa) \ll d(\kappa)$.
2. Let $\text{DMWM} = (\text{DMWM.}\mathsf{setup}, \text{DMWM.}\mathsf{map}, \text{DMWM.}\mathsf{check}, \text{DMWM.}\mathsf{extract})$ be a κ-lossy dual-mode witness map (DMWM) (refer to Definition 7) with tag space $\mathcal{T} = \mathcal{M}$ for the following language:

$$L := \left\{ (s, y) \ : \ \exists\ x \ \mathsf{s.t.}\ y = \text{SPR.}\mathsf{eval}_s(x) \right\}$$

Define the signature scheme $\text{SIG} = (\text{SIG.}\mathsf{setup}, \text{SIG.}\mathsf{keygen}, \text{SIG.}\mathsf{sign}, \text{SIG.}\mathsf{verify})$ as follows:

1. $\text{SIG.}\mathsf{setup}(\kappa)$: On input κ, sample $s \leftarrow \text{SPR.}\mathsf{gen}(\kappa)$. It then samples a random tag $\mathsf{tag} \in \mathcal{T}$, computes $(\mathsf{K}, \mathsf{td}) \leftarrow \text{DMWM.}\mathsf{setup}(\kappa, \ell, \mathsf{tag})$, and discards the trapdoor tk. Set $\mathsf{pub} := (s, \mathsf{K})$.
2. $\text{SIG.}\mathsf{keygen}(\kappa, \mathsf{pub})$: On input the public parameters pub, it samples $x \leftarrow \{0,1\}^{d(\kappa)}$ uniformly at random, and compute $y = \text{SPR.}\mathsf{eval}_s(x)$. Output the signing key $\mathsf{ssk} = x$, and the verification key $\mathsf{vk} = y$.
3. $\text{SIG.}\mathsf{sign}(\mathsf{ssk}, m)$: On input a message m, do the following:
   - Set the tag tag of DMWM to be $\mathsf{tag} = m$.
   - Re-compute the value $y = \text{SPR.}\mathsf{eval}_s(x)$.
   - Generate a representative witness $w^* \leftarrow \text{DMWM.}\mathsf{map}\big(\mathsf{K}, \mathsf{tag}, (s, y), x\big)$, where (s, y) is the statement and x is the corresponding witness.
   - Output the signature $\sigma = w^*$.
4. $\text{SIG.}\mathsf{verify}(\mathsf{vk}, (m, \sigma))$: Parse the signature σ as $\sigma = w^*$. It then sets $\mathsf{tag} = m$ and runs $\text{DMWM.}\mathsf{check}\big(\mathsf{K}, \mathsf{tag}, (s, y), w^*\big)$ to check if the mapping verifies correctly. It outputs 1 if and only if the above verification evaluates to 1.

Fig. 7. Construction of FLTR Signature Scheme SIG

real execution. At this point, all of adversary's signing queries are answered using lossy tags. Moreover, the *extraction* property of DMWM ensures that a forgery under the tag m^* can be efficiently inverted using the trapdoor td. We can use this property to recover a (hopefully second) pre-image of SPR from such a forgery. Note that the extraction property ensures that the extracted value corresponds to valid pre-image x' of y, i.e. $y = \text{SPR.}\mathsf{eval}_s(x')$. Now, we argue that information theoretically the original signing key x still retains enough min-entropy in it, and hence is unpredictable. This follows from the fact that the signing queries are (cumulatively) lossy evaluations of the original signing key x and its derived tampered versions $\{\widetilde{x}_i\}_{i \in [t(\kappa)]}$. In more detail, the κ-lossiness property of DMWM ensures that, even if the adversary observes an arbitrary polynomial number of (lossy) evaluations of the signing key x and its derived tampered versions $\{\widetilde{x}_i\}_{i \in [t(\kappa)]}$ under different tags (which correspond to different messages in our construction), x still has a lot of residual min-entropy. Moreover, the verification key y and the leakage information are too short, and hence the signing key still remains unpredictable to the adversary. (This is formalized using an information-theoretic argument.) Hence, with very high probability the pre-image extracted from the forgery is a second pre-image of y, thereby breaking the security of of SPR.

□

Now, we present the formal proof of Theorem 7.

Proof. Let us denote the adversary for SIG by $\mathcal{F}$. We need to show that no PPT adversary $\mathcal{F}$ is able to come up with a valid forgery in the FLTR signature experiment $\mathsf{Success}^{(\lambda,t)\text{-}\mathsf{FLTR}}_{\Pi,\mathcal{F}}(\kappa)$, except with negligible probability. In order to show this, we define two mental experiments $\mathbf{Game\,G_0}$ and $\mathbf{Game\,G_1}$, and show that the views of the adversary when interacting with these games are computationally indistinguishable. Let us assume that the adversary $\mathcal{F}$ makes at most $t(\kappa)$ tampering queries and $p(\kappa)$ signing queries per tampered key (where $p(\cdot)$ is an arbitrary polynomial in the security parameter κ). We analyze some events within the context of these experiments; events with the same name but different subscripts are analyzed in a similar manner, except that in the scope of the corresponding experiments/games indicated by the subscript.

$\mathbf{Game\,G}_0$: This is identical to the original experiment $\mathsf{Success}^{(\lambda,t)\text{-}\mathsf{FLTR}}_{\Pi,\mathcal{F}}(\kappa)$ as per Definition 13, where the signature scheme SIG is defined as in the original construction. The adversary $\mathcal{F}$ declares the target m^* in the first step. The initial state and the public parameters are sampled as in the real construction and are set as $\mathsf{st} := \mathsf{ssk}$, and $\mathsf{pub} := (s, \mathsf{K})$ respectively. On input the i^{th} tampering function $T_i \in \mathcal{T}$ $(i \in [t(\kappa)])$, the modified signing key $\tilde{x}_i = T_i(x)$ is computed. The answers to the signing query $\{(i, m_j)\}_{i\in[t(\kappa)],j\in[p(\kappa)]}$ to the oracle $\text{SIG}.\mathsf{sign}^*(i,\cdot)$ is computed by the challenger $\mathcal{C}$ as follows: Set the tag tag_j of DMWM to be $\mathsf{tag}_j = m_j$. It then computes $y_i = \text{SPR}.\mathsf{eval}_s(\tilde{x}_i)$ and $\widetilde{w}^j_i \leftarrow \text{DMWM}.\mathsf{map}\big(\mathsf{K}, \mathsf{tag}_j, (s, y_i), \tilde{x}_i\big)$ and outputs the signature $\widetilde{\sigma}_{ij} = \widetilde{w}^j_i$ on the j^{th} message with respect to the i^{th} tampered key. On input a leakage query $f_j(\cdot) : \{0,1\}^* \to \{0,1\}^{\lambda_j}$, the challenger $\mathcal{C}$ returns $f_j(\mathsf{st})$ to $\mathcal{F}$, provided the cumulative leakage received by $\mathcal{F}$ does not exceed the bound $\lambda(\kappa)$.

Let Succ_0 denote the following event: The forger $\mathcal{F}$ is able to produce a "valid" forgery, i.e., $\mathcal{F}$ produces $\big(m^*, \sigma^* = w^*\big)$ such that $\text{SIG}.\mathsf{verify}(\mathsf{vk}, (m^*, \sigma^*)) = 1$ and m^* is never submitted to the signing oracle. From the definition of game $\mathbf{G}_0$, it follows that:

$$\Pr[\mathsf{Succ}_0] = \Pr\left[\mathsf{Success}^{(\lambda,t)\text{-}\mathsf{FLTR}}_{\Pi,\mathcal{F}}(\kappa)\right]$$

$\mathbf{Game\,G}_1$: In this game, we change the way the public parameters are sampled using the setup algorithm $\text{SIG}.\mathsf{setup}$. The adversary/forger $\mathcal{F}$ first declares the target message m^*. We now change the way the string K of DMWM is generated. In particular, instead of sampling a random tag $\mathsf{tag}^* \in \mathcal{M}$, the challenger sets the tag $\mathsf{tag}^* = m^*$ to generate $(\mathsf{K}, \mathsf{td}) \leftarrow \text{DMWM}.\mathsf{setup}(\kappa, \ell, \mathsf{tag}^*)$. Also, instead of discarding the inversion trapdoor td (as in the real construction), the challenger retains the td. The way the other public parameters are sampled and the answers to the signing queries are given remain identical to the previous game.

Claim. $\big|\Pr[\mathsf{Succ}_1] - \Pr[\mathsf{Succ}_0]\big| \leq \mathsf{negl}(\kappa)$

Proof. The proof of the above claim follows from the *hidden tag* property of $\text{DMWM} = (\text{DMWM}.\mathsf{setup}, \text{DMWM}.\mathsf{map}, \text{DMWM}.\mathsf{check}, \text{DMWM}.\mathsf{extract})$. In particular, any distinguisher between Game 0 and Game 1 can be used to build another distinguisher $\mathcal{D}_{\text{DMWM}}$ for DMWM. Also, let us denote by tag_0 the tag tag (used in Game 0) and by tag_1 the tag m^*. The distinguisher $\mathcal{D}_{\text{DMWM}}$ simulates the environment for $\mathcal{F}$ as follows:

– Receives the message m^* from the adversary $\mathcal{F}$, and sets the tag $\mathsf{tag}_1 = m^*$. It also samples another random tag $\mathsf{tag}_0 \in \mathcal{M}$. It then forwards both these tags tag_0 and tag_1 to its challenger and receives the tuple $(\mathsf{K}, \mathsf{st})$ corresponding to one of these tags. It then computes the public parameters as: sample the function index $s \leftarrow \text{SPR.}\mathsf{gen}(\kappa)$, and sets $\mathsf{pub} = (s, \mathsf{K})$.
– Sample $x \leftarrow \{0,1\}^{d(\kappa)}$, and compute $y = \text{SPR.}\mathsf{eval}_s(x)$. Set $\mathsf{ssk} = x$, and $\mathsf{vk} = y$. It then returns (pub, y) to $\mathcal{F}$. It also sets $\mathsf{st} := \{\mathsf{ssk}\}$, and initializes a list $\mathcal{L} := 0$.
– On input a leakage query $f_j(\cdot) : \{0,1\}^* \to \{0,1\}^{\lambda_j}$, check if $\mathcal{L} + |f_j(\mathsf{st})| \leq \lambda(\kappa)$, if so, it returns $f_j(\mathsf{st})$ to $\mathcal{F}$, and sets $\mathcal{L} := \mathcal{L} + |f_j(\mathsf{st})|$; else, it aborts. On input a tampering query $T_i \in \mathcal{T}$, it checks if $i \in [t]$, if so, it computes $\widetilde{x}_i := T_i(x)$, else it aborts.
– Upon input the message (i, m_j) (for some message $m_j \in \mathcal{M}$), check if $i \notin [t]$, if so, return $\perp$. Otherwise, $\mathcal{D}_{\text{DMWM}}$ proceeds as follows: set the tag $\mathsf{tag}_j = m_j$, compute $y_i = \text{SPR.}\mathsf{eval}_s(\tilde{x}_i)$, and $\widetilde{w}_i^j \leftarrow \text{DMWM.}\mathsf{map}\big(\mathsf{K}, \mathsf{tag}_j, (s, y_i), \tilde{x}_i\big)$. Finally, it outputs the signature $\widetilde{\sigma}_{ij} = \widetilde{w}_i^j$.
- Output whatever $\mathcal{F}$ outputs.

For the analysis, note that, the two games Game 0 and Game 1 differ in how the public parameters are sampled. If the tag tag_0 was used for sampling the function index, this exactly corresponds to Game 0. This is true in our case, since the tag tag_0 was sampled randomly by $\mathcal{D}_{\text{DMWM}}$. On the other hand, if the tag tag_1 was used, this corresponds to Game 1. Hence, if the advantage of $\mathcal{F}$ in distinguishing these two games is non-negligible, the adversary $\mathcal{D}_{\text{DMWM}}$ also breaks the hidden tag property of DMWM with non-negligible probability, thus yielding a contradiction. □

Note that, at this point all of the adversary's signing queries correspond to messages $m \neq m^*$, and hence correspond to lossy tags. Hence the signatures are essentially "*lossy evaluations*" of the secret signing key x and its derived tampered versions $\{\widetilde{x}_i\}_{i \in [t(\kappa)]}$.

Let Ext_1 denote the following event: The forger $\mathcal{F}$ is able to produce a "valid" forgery, and it is possible to recover a "valid" pre-image from the forgery, i.e., if $\mathcal{F}$ produces $\big(m^*, \sigma^* = w^*\big)$ such that $\text{SIG.}\mathsf{verify}(\mathsf{vk}, (m^*, \sigma^*)) = 1$, and m^* was never queried to the signing oracle, then the output $x^* = \mathsf{extract}(\mathsf{td}, x, w^*)$ is such that $\text{SPR.}\mathsf{eval}_s(x^*) = y$.

Claim. $\big|\ \Pr[\mathsf{Ext}_1] - \Pr[\mathsf{Succ}_1]\ \big| \leq \mathsf{negl}(\kappa)$

Proof. The claim follows from the extraction property of DMWM. In particular, the tag $\mathsf{tag}^*(= m^*)$ is extractable and hence we can run the extractor $\mathsf{extract}$ of DMWM to extract a pre-image x^* from the representative witness w^* (created using the tag tag^*). Further, the extraction property of DMWM ensures that the extracted value x^* is a valid witness corresponding to the statement (s, y), except with negligible probability. □

Define the event $\mathsf{SameExt}_1$ to be event that Ext_1 happens, and in addition the extracted value is *same* as the original signing key value, i.e, $x^* = \mathsf{extract}(\mathsf{td}, x, w^*)$ occurs, where $(m^*, \sigma^* = w^*)$ is the forgery produced by $\mathcal{F}$, and additionally $x^* = x$.

Claim. $\big|\Pr[\mathsf{SameExt}_1] - \Pr[\mathsf{Ext}_1]\big| \leq \mathsf{negl}(\kappa)$

Proof. The claim follows from the *second pre-image resistance* property of $\textsc{spr} = (\textsc{spr}.\mathsf{gen}, \textsc{spr}.\mathsf{eval})$. That is, if Ext_1 happens but not $\mathsf{SameExt}_1$, then we run Game $\mathbf{G}_1$ with SPR challenge (s, x) to recover a pre-image $x^* \neq x$ such that $\textsc{spr}.\mathsf{eval}_s(x) = \textsc{spr}.\mathsf{eval}_s(x^*)$, thus breaking the second pre-image property of $\textsc{spr}.\mathsf{eval}$ with non-negligible probability. □

Recall that, at the end of Game $\mathbf{G}_1$, all of adversary's signing responses correspond to lossy tags.

From this point onward, all our arguments will be solely *information-theoretic*. To complete the proof we need to show that $\Pr[\mathsf{SameExt}_1]$ is negligible.

Claim. $\Pr[\mathsf{SameExt}_1] \leq \mathsf{negl}(\kappa)$

Proof. Note that, the information about x that is revealed to $\mathcal{F}$ comes from the following values: i) the verification key y corresponding to x, ii) the leakage queries asked by $\mathcal{F}$, iii) the signatures $\widetilde{w}_i^j$ $(i \in [t(\kappa)],\ j \in [p(\kappa)])$. The verification key in our construction reveals at most m bits of information about x. The leakage queries asked by $\mathcal{F}$ reveal at most λ bits of information about x. Let us now estimate the information revealed about x by the signatures $\widetilde{w}_i^j$. Note that, in our construction

$$\widetilde{w}_i^j = \textsc{dmwm}.\mathsf{map}\big(\mathsf{K}, \mathsf{tag}, (s, y_i), \widetilde{x}_i\big) = \mathsf{expand}_{\mathsf{K}}(\mathsf{compress}_{\mathsf{K},(s,y_i)}(\widetilde{x}_i), \mathsf{tag}),$$

where $\mathsf{compress}_{\mathsf{K},(s,y_i)} : \{0,1\}^* \to S^{(i)}$, where $S^{(i)} = S_{\mathsf{K},(s,y_i)}$ and $|S_{\mathsf{K},(s,y_i)}| \leq 2^\kappa$ (since $\textsc{dmwm}$ is κ-lossy). Hence, the tampering queries of $\mathcal{F}$ corresponds to sets $\{S^{(i)}\}_{i\in[t]}$ which collectively reveals at most $\kappa t(\kappa)$ bits of information about x. Moreover, the signing queries (i, m) with $i = 0$ (i.,e, with respect to the untampered key x) also reveals at most κ bits of information about x.

Let us denote the view of the forger $\mathcal{F}$ in Game $\mathbf{G}_1$ as $\mathsf{View}_{\mathcal{F},1}$. We also denote by $\mathcal{L}$ the random variable representing the leakage information got by $\mathcal{F}$. Below, we formally show that x still possesses a high residual min-entropy, even conditioned on $\mathsf{View}_{\mathcal{F},1}$.

$$\begin{aligned}
\widetilde{\mathrm{H}}_\infty\big(x \mid \mathsf{View}_{\mathcal{F},1}\big) &= \widetilde{\mathrm{H}}_\infty\big(x \mid (y, \mathcal{L}, \{\widetilde{w}_i^j\}_{i\in[t(\kappa)],j\in[p(\kappa)]})\big) \\
&\geq \widetilde{\mathrm{H}}_\infty\big(x \mid (y, \{\widetilde{w}_i^j\}_{i\in[t(\kappa)],j\in[p(\kappa)]})\big) - \lambda(\kappa). \\
&\geq \widetilde{\mathrm{H}}_\infty(x \mid y) - \lambda(\kappa) - \kappa(t(\kappa)+1). \\
&\geq \widetilde{\mathrm{H}}_\infty(x) - \lambda(\kappa) - \kappa(t(\kappa)+1) - m(\kappa). \\
&\geq \omega(\log \kappa)
\end{aligned}$$

The last line of the above equation follows from our choice of parameters, i.e., $d(\kappa) > \lambda(\kappa)+\kappa(t(\kappa)+1)+m(\kappa)+\omega(\log\kappa)$. Hence the min-entropy $\widetilde{\mathrm{H}}_\infty(x|\mathsf{View}_{\mathcal{F},1})$ of x condition on $\mathsf{View}_{\mathcal{F},1}$ is at least $\omega(\log\kappa)$. This completes the proof of the above claim. □

Putting together all the claims completes the proof of Theorem 7. □

6 Extension to Continuous Leakage and Tampering

In this section, we show how to extend our construction of fully leakage and tamper-resilient signatures from Section 5 to the setting of continuous leakage and tampering attacks. However, as shown by Gennaro et al. [26] it is *impossible* to construct any cryptographic primitive allowing an arbitrary number of unrestricted tampering functions. There are two main ways to bypass this impossibility result: (1) using a **self-destruct** mechanism [26], meaning that when tampering is detected, the cryptographic device can erase all internal data, so that an adversary cannot obtain anything more from the device, and (2) using a **key-update** mechanism, i.e., allow the device to refresh its secret state (key) using fresh randomness [34] between periods of tampering and leakage.

Kalai, Kanukurthi and Sahai [34] gave the first construction of a signature scheme resilient to continual leakage and tampering (CLT) attacks, assuming (1) the signature scheme is allowed to have a *key-update* mechanism and (2) the tampering is *persistent*, i.e., the tampering is applied to the current version of the secret that may have been overwritten by the previous tampering queries. They also gave a construction of a signature scheme resilient to continuous tampering and bounded leakage (CTBL) without a key-update procedure, but assuming that the device has a self-destruct capability. Note that, the model of tampering that we consider in this work is *non-persistent* tampering, where the tampering functions are always applied to the original secret. If a signature scheme *does not* have a key-update mechanism, non-persistent tampering is *stronger* than persistent tampering. In fact, a recent work [24] showed that, it is **impossible** to construct a signature scheme resilient to *continuous non-persistent* tampering attacks, if it *does not* have a key-updating mechanism (even if it can self-destruct). The impossibility result extends even if the signature scheme has a key-update mechanism, but the update is run *only* when tampering is detected.

The above impossibility result, however, does not rule the possibility of constructing continuous (non-persistent) tamper-resilient signature scheme. In fact, the impossibility result only hold for signature schemes that *do not* have a key-update mechanism or update the key *only* when tampering is detected. We consider the (im)possibility of designing signature schemes resilient to CTL attacks if the device can update its signing key periodically, regardless of whether tampering is detected or not. Our main observation is that, indeed, we can construct such a signature scheme with periodic key updates, thereby bypassing the impossibility result of [24]. Our construction uses a rather straight-forward extension of our basic construction from Section 5. To this end, we first abstract out the SPR function as an instance of a noisy leakage-resilient one-way relation (LR-OWR). Then, we present a *generalization* of (noisy) LR-OWR to allow for continuous leakage attacks. Our definition of (noisy) continuous leakage-resilient one-way relation (CLR-OWR) follows closely along the lines of Dodis et al. [16]. Before, describing CLR-OWR, we first recall the definition of LR-OWR.

6.1 One-way Relation Resilient to Leakage Attacks

In this section, we define one way relations resilient to leakage attacks. Informally, a LR-OWR function ($\text{OWR.}\mathsf{gen}, \text{OWR.}\mathsf{eval}, \text{OWR.}\mathsf{ver}$) guarantees the following: Given an image $y = \text{OWR.}\mathsf{eval}_s(x)$ for a random x in the domain and $s \leftarrow \text{OWR.}\mathsf{gen}(1^\kappa)$, it is hard for an adversary to produce any pre-image x' of y under $\text{OWR.}\mathsf{eval}$, even if the adversary obtains λ bits of leakage on x. We call such a LR-OWR, from n-bit inputs to m-bit outputs, a $(\lambda; n, m)$-leakage-resilient one-way relation ($(\lambda; n, m)$-LR-OWR).

For our application to LTR signatures, we will need to rely on a stronger form leakage resilience, namely, the *noisy leakage* model [16]. Noisy leakage is a generalization of bounded leakage, where the adversary can learn functions with arbitrarily large output-lengths, as long as the entropy of the secret-key does not decrease significantly. Following [16] we define the leakiness of a function as follows:

Definition 14 (ℓ-leaky functions). *A probabilistic leakage function $f : \{0,1\}^* \to \{0,1\}^*$ is ℓ-leaky if there exists some (possibly inefficiently computable) function f' such that:*

- *For all $x \in \{0,1\}^*$, $f(x) \approx_s f'(x)$ (over the randomness of f and f').*
- *For all integers $n \geq 1$, we have that $\widetilde{\mathrm{H}}_\infty(U_n | f'(U_n)) \geq n - \ell$, where U_n is the uniform distribution over $\{0,1\}^n$.*

It can be easily shown that a function whose output length is bounded by ℓ-bits is ℓ-leaky. We now define a noisy leakage-resilient one-way relation (LR-OWR).

Definition 15 (Noisy LR-OWR). *We say that ($\text{OWR.}\mathsf{gen}, \text{OWR.}\mathsf{eval}, \text{OWR.}\mathsf{ver}$) is a $(\lambda; n, m)$-noisy leakage-resilient one-way relation* ($(\lambda; n, m)$-noisy LR-OWR) *if it satisfies the following correctness and security properties:*

Correctness. For all $s \leftarrow \text{OWR.}\mathsf{gen}(1^\kappa)$ and all $x \xleftarrow{\$} \{0,1\}^{n(\kappa)}$, if $y = \text{OWR.}\mathsf{eval}_s(x)$, then $\text{OWR.}\mathsf{ver}(x, y) = 1$, and $y \in \{0,1\}^{m(\kappa)}$.

Security. For any PPT adversary $\mathcal{A}_{\mathsf{lr\text{-}owr}}$, we have $\mathsf{Pr}[\mathcal{A}_{\mathsf{lr\text{-}owr}} \text{ wins}] \leq \mathsf{negl}(\kappa)$ in the following game:

- The challenger chooses $s \leftarrow \text{OWR.}\mathsf{gen}(1^\kappa)$, samples $x \xleftarrow{\$} \{0,1\}^{n(\kappa)}$ and computes $y = \text{OWR.}\mathsf{eval}_s(x)$. It then gives y to $\mathcal{A}_{\mathsf{lr\text{-}owr}}$.
- The adversary $\mathcal{A}_{\mathsf{lr\text{-}owr}}$ may provide the description of an efficiently computable leakage function $f(\cdot)$, provided that f is λ-*leaky*.
- The adversary wins if it produces a value x^* such that the following holds: i) $\text{OWR.}\mathsf{ver}(x^*, y) = 1$, and ii) the leakage query f made by $\mathcal{A}_{\mathsf{ltr\text{-}owr}}$ is λ-leaky.

We may abbreviate $(\lambda, t; n, m)$-noisy LR-OWR as λ-noisy LR-OWR when (n, m) are not relevant or are clear from the context.

It was shown in [17] that a second pre-image resistant (SPR) function with n bits input and m bits output is a λ-noisy LR-OWR with $\lambda \leq n - m - \omega(\log \kappa)$.

FLTR signatures from (noisy) LR-OWRs. In this section, we explain our construction of FLTR signature (see Section 5) in a slightly more general way. Recall that, our construction was based on a SPR function $\text{SPR} = (\text{SPR.gen}, \text{SPR.eval})$, where $y = \text{SPR.eval}_s(x)$ was the verification key and x is the secret signing key. To sign a message m, we set the tag of the DMWM to be the message m, and construct a representative witness w^* for the statement: $\exists x, y = \text{SPR.eval}_s(x)$ using x as the original witness. In general, we do not need a SPR function for the above construction; any λ-noisy LR-OWR $(\text{OWR.gen}, \text{OWR.eval}, \text{OWR.ver})$ will suffice.

Theorem 8. *Let* $(\text{OWR.gen}, \text{OWR.eval}, \text{OWR.ver})$ *be a* $\lambda(\kappa)$*-noisy LR-OWR with input size* $d(\kappa)$ *and output size* $m(\kappa)$*, and let* DMWM *be a* κ*-lossy DMWM with tag space* $\mathcal{T} = \mathcal{M}$*. Then, the modified construction sketched above is a* $\big(\lambda'(\kappa), t(\kappa)\big)$-FLTR *signature scheme in the bounded leakage model, as long as* $\lambda'(\kappa) \leq \lambda(\kappa) - \kappa(t+1)$*.*

The proof of security is very similar to the original proof of Theorem 7. In particular, when we switch from game G_0 to game G_1, all of adversaries' (tampered) signing queries correspond to lossy tags while the forgery being extractable. Hence, the entire view of the adversary ((tampered) signing and leakage queries) does not reduce the entropy of x significantly. Hence, we can think of the view of the adversary as entropy-bounded leakage (but not length-bounded unless we could efficiently compress it). Note that, the tampering queries made by the adversary is simulated by the reduction with access to only the leakage oracle (so called *leakage-to-tamper* reduction). Successful forgery in this setting means the adversary only receives entropy-bounded leakage on x, but still manages to produce (a representative witness w^* of) some pre-image x^*, thus breaking one-way security.

Although the above generalization does not seem significant at first, this view of our basic construction will make it easier to extend to the setting of continuous leakage, which we will do in the next section.

6.2 One-way Relation Resilient to Continuous Leakage Attacks

We now present the generalization of LR-OWR to the setting of continuous leakage attacks. We still consider the notion of noisy leakage from above.

Definition 16 (Noisy CLR-OWR). *We say that* $(\mathsf{gen}_{\mathsf{clr\text{-}owr}}, \mathsf{refresh}, \mathsf{ver}_{\mathsf{clr\text{-}owr}})$ *is a* λ*-noisy* continuous-leakage-resilient *one-way relation* (λ-noisy CLR-OWR) *if it satisfies the following correctness and security properties:*

Correctness. For any polynomial $p = p(n)$, if we sample $(y, x) \leftarrow \mathsf{gen}_{\mathsf{clr\text{-}owr}}(1^\kappa; r)$, $x_1 \leftarrow \mathsf{refresh}(x; r_1), \cdots, x_p \leftarrow \mathsf{refresh}(x_{p-1}; r_p)$, then with very high probability $\mathsf{ver}_{\mathsf{clr\text{-}owr}}(y, x) = \mathsf{ver}_{\mathsf{clr\text{-}owr}}(y, x_1) = \cdots = \mathsf{ver}_{\mathsf{clr\text{-}owr}}(y, x_p)$.

Security. For any PPT adversary $\mathcal{A}_{\mathsf{clr\text{-}owr}}$, we have $\mathsf{Pr}[\mathcal{A}_{\mathsf{clr\text{-}owr}} \text{ wins}] \leq \mathsf{negl}(\kappa)$ in the following game:

– The challenger samples $(y, x) \leftarrow \mathsf{gen}_{\mathsf{clr\text{-}owr}}(1^\kappa; r)$, and gives y to $\mathcal{A}_{\mathsf{clr\text{-}owr}}$.

– The adversary $\mathcal{A}_{\mathsf{clr\text{-}owr}}$ may run for arbitrarily many leakage rounds. In each round (say round i):
  - The adversary $\mathcal{A}_{\mathsf{clr\text{-}owr}}$ may provide the description of an efficiently computable leakage function $f_i(\cdot)$, provided that f_i is λ-leaky.
  - At the end of the round i, the challenger samples $x_{i+1} \leftarrow \mathsf{refresh}(x_i; r_i)$ and updates $x := x_{i+1}$.
– The adversary wins if it produces a value x^* such that the following holds: (i) $\mathsf{Ver}_{\mathsf{clr\text{-}owr}}(y, x^*) = 1$, and (ii) the leakage queries f_i made by $\mathcal{A}_{\mathsf{clr\text{-}owr}}$ in each round must be at most λ-leaky.

The work of [16] constructs a noisy CLR-OWR based on the k-Linear assumption.

Remark 4. One may consider a stronger notion of (λ, μ)-noisy CLR security, where the adversary can observe up to λ entropy-bounded leakage from each secret key x_i *in between* the refresh operations, and additionally leak up to μ bits of leakage on the internal state $\mathsf{state}_i = (x_i, r_i)$ used *during* each refresh operation, i.e., $x_{i+1} = \mathsf{refresh}(x_i; r_i)$. This is called the "leakage-of-refreshing security". The CLR-OWR of [16] achieves the leakage-of-refreshing security.

Theorem 9. [11, 16]. *For any polynomial $\lambda(\cdot)$ and any constant $\epsilon > 0$, there exists $\lambda(\kappa)$-noisy CLR-OWR where the relative leakage rate (ratio of leakage to secret key size) is $\lambda/|x| = 1-\epsilon$ under the linear assumption in bilinear maps. Under the same assumption and for any polynomial $\lambda(\cdot)$, there also exists a $\lambda(\kappa)$-noisy CLR-OWR with the stronger "leakage-of-refreshing security", where, for some constant $c > 0$, the relative leakage rate is $\lambda/|x| > c$.*

6.3 FLTR signatures in the continuous leakage and tampering model.

6.3.1 Security Model. In this section, we present the security model for FLTR signatures in the setting of continuous leakage and tampering (CLT) attacks. This is an extension of the definition of FLTR signatures in the bounded leakage model (see Section 5.1).

Definition 17. ((λ, t)-FLTR security–CLT model). *We say that a signature scheme* $\mathrm{SIG} = (\mathrm{SIG.}\mathsf{setup}, \mathrm{SIG.}\mathsf{keygen}, \mathrm{SIG.}\mathsf{refresh}, \mathrm{SIG.}\mathsf{sign}, \mathrm{SIG.}\mathsf{verify})$ *is* (λ, t)-fully-leakage and tamper-resilient (FLTR) *in the* continuous leakage and tampering (CLT) *model if for all PPT adversaries/forgers $\mathcal{F}$ there exists a negligible function $\mathsf{negl} : \mathbb{N} \to \{0, 1\}$ such that* $\Pr\left[\mathsf{Success}^{(\lambda,t)\text{-}\mathsf{CFLTR}}_{\Pi,\mathcal{F}}(\kappa)\right] \leq \mathsf{negl}(\kappa)$, *where the event* $\mathsf{Success}^{(\lambda,t)\text{-}\mathsf{CFLTR}}_{\Pi,\mathcal{F}}(\kappa)$ *is defined via the following experiment between a challenger $\mathcal{C}$ and the forger $\mathcal{F}$:*

1. *Initially, the challenger $\mathcal{C}$ computes* $\mathsf{pub} \leftarrow \mathrm{SIG.}\mathsf{setup}(\kappa)$ *and* $(\mathsf{ssk}, \mathsf{vk}) \leftarrow \mathrm{SIG.}\mathsf{keygen}(\kappa, \mathsf{pub})$, *and sets* $\mathsf{st} = \{\mathsf{ssk}\}$, *and* $L = 0$.
2. *The forger (on receiving* pub *and* vk*) runs for arbitrarily many leakage rounds $i = 1, 2, \cdots$. In each round i, it can adaptively query the following oracles as defined below:*

- **Signing queries**: *The signing oracle* $\text{SIG.sign}^*_{\mathsf{ssk}_i}(\cdot)$ *receives as input a message* $m_j \in \mathcal{M}$*, where* j *is an arbitrary polynomial in the security parameter. The challenger* $\mathcal{C}$ *then samples* $r_j \leftarrow \mathcal{R}$*, and computes* $\sigma_j \leftarrow \text{SIG.sign}(\mathsf{ssk}_i, m_j, r_j)$*. Set* $\mathsf{st} := \mathsf{st} \cup \{r_j\}$ *and outputs* σ_j*.*
- **Leakage queries**: *The leakage oracle receives as input (the description of) an efficiently computable function* $f_j : \{0,1\}^* \to \{0,1\}^{\lambda_j}$*. If* $|L| + \lambda_j \leq \lambda(\kappa)$*, it responds with* $f_j(\mathsf{st})$*. Otherwise, it outputs* $\perp$*.*
- **Key-refresh queries**: *On a key-refresh query, the challenger samples a fresh randomness* r_i*. The next-round secret key is then sampled as* $\mathsf{ssk}_{i+1} \leftarrow \text{SIG.refresh}_{\mathsf{vk}}(\mathsf{ssk}; r_i)$*. Set* $\mathsf{ssk} = \mathsf{ssk}_{i+1}$*, and reset* $\mathsf{st} = \{\mathsf{ssk}\}$ *and* $L = 0$*.*
- **Tampering queries**: *The forger can adaptively ask at most* $t(\kappa)$ *tampering queries in each round. For each tampering query* T_j *(where* $j \in [t(\kappa)]$*), the challenger computes* $(\widetilde{\mathsf{ssk}}_j, \widetilde{r}_j) = T_j(\mathsf{st})$*. Subsequently,* $\mathcal{F}$ *can adaptively query the tampered-signing oracle* $\text{SIG.sign}(\widetilde{\mathsf{ssk}}_j, \cdot, \widetilde{r}_j)$ *using messages in* $\mathcal{M}$*.*

3. Eventually, $\mathcal{F}$ *outputs a message-signature pair* (m^*, σ^*) *as the purported forgery.*

$\mathsf{Success}^{(\lambda,t)\text{-CFLTR}}_{\Pi,\mathcal{F}}(\kappa)$ *denotes the event in which the following happens:*

- *The signature* σ^* *verifies with respect to the original verification key* vk*, i.e.,* $\text{SIG.verify}(\mathsf{vk}, (m^*, \sigma^*)) = 1$*.*
- m^* *was never queried as input to the signing or tampered signing oracle by the forger* $\mathcal{F}$ *at any round.*
- *The output length of all the leakage functions* $\sum_j \lambda_j$ *in each round is at most* $\lambda(\kappa)$*.*
- *The number of tampering queries made by* $\mathcal{F}$ *in each round is at most* $t(\kappa)$*.*

Remark 5. As before, we also consider a selective variant of the above definition, where the message m^* (on which the forgery is to be produced) is declared by the adversary before receiving the public parameters pub and the verification key vk. We call this *selectively unforgeable* (λ, t)-FLTR signature scheme in the CLT model.

Remark 6. We also consider a stronger variant of the definition that provides *leakage-of-refreshing* security, by modifying the challenger so that, during a key-refresh query, it sets $\mathsf{st} = \mathsf{ssk}||r$ to include the new secret key ssk and the randomness r used during the refreshing. However, we note that, one cannot allow for tampering the randomness used in the key refresh procedure. We explain this in more detail below.

Impossibility of FLTR signatures with tamperable randomness during updates. One may be instigated to extend the above definition of (λ, t)-FLTR security in the CLT model to the setting where the adversary may also tamper with the randomness used during the key refresh operations. In particular, during the update phase from round i to round $i+1$, the adversary may submit tampering functions T and get back the (tampered) updated key $\mathsf{ssk}_{i+1} = \text{SIG.refresh}_{\mathsf{vk}}(\widetilde{\mathsf{ssk}_i}, \widetilde{r_i})$, where $(\widetilde{\mathsf{ssk}_i}, \widetilde{r_i}) = T(\mathsf{ssk}_i, r_i)$.

However, we observe that achieving such a notion of security is *impossible*. This is because, the adversary can reset the randomness r_i used in each round of the refresh operation to be the all-zero string and make the update process deterministic. At this point, the adversary may launch a *key pre-computation* attack, where the adversary (via leakage queries) may request bits of some future secret key, one by one, in earlier rounds. This is also the reason why one cannot have a deterministic update process while designing any continuous leakage-resilient primitive.

6.3.2 Construction of FLTR signature in the CLT model. Given a λ-noisy CLR-OWR, we can easily generalize the construction of our signature scheme from Section 5 to the continuous leakage and tampering setting. We now present our construction of FLTR signatures in the CLT model (see Figure 8).

1. Let $(\mathsf{gen}_{\mathsf{clr\text{-}owr}}, \mathsf{refresh}, \mathsf{ver}_{\mathsf{clr\text{-}owr}})$ be a λ-noisy continuous-leakage-resilient one-way relation (λ-noisy CLR-OWR) from $\{0,1\}^{d(\kappa)}$ to $\{0,1\}^{m(\kappa)}$, where $m(\kappa) \ll d(\kappa)$.
2. Let $\text{DMWM} = (\text{DMWM.}\mathsf{setup}, \text{DMWM.}\mathsf{map}, \text{DMWM.}\mathsf{check}, \text{DMWM.}\mathsf{extract})$ be a κ-lossy dual-mode witness map (DMWM) (refer to Definition 7) with tag space $\mathcal{T} = \mathcal{M}$ for the following language:

$$L := \{(s,y) : \exists\, x \text{ s.t. } \mathsf{ver}_{\mathsf{clr\text{-}owr}}(y,x) = 1\}$$

Define the signature scheme $(\text{SIG.}\mathsf{setup}, \text{SIG.}\mathsf{keygen}, \text{SIG.}\mathsf{refresh}, \text{SIG.}\mathsf{sign}, \text{SIG.}\mathsf{verify})$ as follows:

1. $\text{SIG.}\mathsf{setup}(\kappa)$: On input κ, sample a random tag $\mathsf{tag} \in \mathcal{T}$, computes $(\mathsf{K}, \mathsf{td}) \leftarrow \text{DMWM.}\mathsf{setup}(\kappa, \ell, \mathsf{tag})$, and discards the trapdoor tk. Set $\mathsf{pub} := \mathsf{K}$.
2. $\text{SIG.}\mathsf{keygen}(\kappa, \mathsf{pub})$: On input the public parameters pub, it samples $(y,x) \leftarrow \mathsf{gen}_{\mathsf{clr\text{-}owr}}(1^\kappa)$. Output the signing key $\mathsf{ssk} = x$, and the verification key $\mathsf{vk} = y$.
3. $\text{SIG.}\mathsf{refresh}_{\mathsf{vk}}(\mathsf{ssk})$: Parse $\mathsf{vk} = y$. Sample randomness r and run $\mathsf{ssk}' \leftarrow \mathsf{refresh}_y(\mathsf{ssk}; r)$.
4. $\text{SIG.}\mathsf{sign}(\mathsf{ssk}, m)$: On input a message m, do the following:
   - Set the tag tag of DMWM to be $\mathsf{tag} = m$.
   - Generate a representative witness $w^* \leftarrow \text{DMWM.}\mathsf{map}\big(\mathsf{K}, \mathsf{tag}, (s,y), x\big)$, where (s,y) is the statement and x is the corresponding witness.
   - Output the signature $\sigma = w^*$.
5. $\text{SIG.}\mathsf{verify}(\mathsf{vk}, (m,\sigma))$: Parse the signature σ as $\sigma = w^*$. It then sets $\mathsf{tag} = m$ and runs $\text{DMWM.}\mathsf{check}\big(\mathsf{K}, \mathsf{tag}, (s,y), w^*\big)$ to check if the mapping verifies correctly. It outputs 1 if and only if the above verification evaluates to 1.

Fig. 8. Construction of FLTR Signature Scheme in the CLT model

Theorem 10. *Suppose $\lambda(\kappa)$, $t(\kappa)$, $d(\kappa)$ and $m(\kappa)$ are parameters such that there exist a $\big((\lambda(\kappa)+1); d(\kappa), m(\kappa)\big)$-noisy* CLR-OWR*, a κ-lossy DMWM with tag sapce $\mathcal{T} = \mathcal{M}$ (where $\mathcal{M}$ is the message sape of the signature scheme). Then there exists a selectively-secure $(\lambda(\kappa), t(\kappa))$-*FLTR *signature scheme, as long as the parameters satisfy:*

$$0 \le \lambda(\kappa) \le d(\kappa) - \kappa\big(t(\kappa)+1\big) - m(\kappa) - \omega(\log \kappa).$$

Further, if the noisy CLR-OWR *satisfies* leakage-of-refreshing *security, then so does the resulting signature scheme.*

Proof. The main idea of the proof is similar to the proof of Theorem 7. We will refer to that proof, highlighting the main technical differences. Suppose there is an adversary $\mathcal{F}$ who can break the security of the above signature scheme. We use this adversary to break the security of the the noisy CLR-OWR. Similar to the proof of Theorem 7, we define Game G_0 to be the original game corresponding to the FLTR security experiment (in the CLT model), and Game G_1 to be the game where we set the tag $\mathsf{tag}^* = m^*$ (note that m^* is given by the adversary as part of the selective unforgeability game). As argued before, the *hidden tag* property of DMWM ensures that both these experiments are indistinguishable. We then define the event Ext_1 as follows: The forger $\mathcal{F}$ is able to produce a "valid" forgery, and it is possible to recover a "valid" pre-image from the forgery, i.e., if $\mathcal{F}$ produces $\left(m^*, \sigma^* = w^*\right)$ such that $\text{SIG}.\mathsf{verify}(\mathsf{vk}, (m^*, \sigma^*)) = 1$, and m^* was never queried to the signing oracle, then the output $x^* = \mathsf{extract}(\mathsf{td}, x, w^*)$ is such that $\mathsf{ver}_{\mathsf{clr\text{-}owr}}(y, x^*) = 1$.

As in the proof of Theorem 7, the event Ext_1 happens with a noticeable probability. We now show how to use the forger $\mathcal{F}$ breaking Game G_1 to break the security of the CLR-OWR. The reduction $\mathcal{B}$ samples $(\mathsf{K}, \mathsf{td})$ as in Game G_1. Recall that the forger $\mathcal{F}$ expects to run in many epochs (periods between issuing a key refresh query). The view of $\mathcal{F}$ during each epoch i consists of his random coins together with the (tampered) signing queries and leakage queries issued during that epoch. The main idea is that the reduction $\mathcal{B}$ can simulate this view for $\mathcal{F}$ by learning a single leakage-function g_i on the secret key x_i of the CLR-OWR in each epoch. The selection of g_i (described below) will ensure that:

1. The simulation perfectly matches Game G_1. In particular, the event Ext_1 occurs with a non-negligible probability.
2. If the event Ext_1 occurs, then every function g_i queried by $\mathcal{B}$ is at most $(\lambda(\kappa) + 1)$-leaky.

When the event Ext_1 occurs, then $\mathcal{B}$ can extract $x^* = \mathsf{extract}(\mathsf{td}, x, w^*)$ such that $\mathsf{ver}_{\mathsf{clr\text{-}owr}}(y, x^*) = 1$, thereby winning the CLR-OWR security game. Therefore, the two requirements (1) and (2) ensure that this occurs with a non-negligible probability, which leads to a contradiction.

We are left to describe how $\mathcal{B}$ chooses the leakage functions so as to satisfy the above two conditions. In epoch i, the function $g_i : \{0,1\}^* \to \{0,1\}^*$ includes, in its description, the entire view (including the random coins) of the forger $\mathcal{F}$ up to the start of epoch i, along with the verification key vk of the signature scheme. The function $g_i(x_i)$ first checks whether $\mathsf{ver}_{\mathsf{clr\text{-}owr}}(y, x_i) = 1$ and, if not, returns a 0. Otherwise, it internally runs the code of $\mathcal{F}$ for that epoch, and uses the current secret key x_i (and internal random coins) to answer the leakage queries and the (tampered) signing queries. The output of g_i consists of all the answers to the various queries asked by $\mathcal{F}$ during the epoch.

It is easy to see that this leakage can be used by $\mathcal{B}$ to (perfectly) simulate the epoch to $\mathcal{F}$, so we satisfy the requirement (1). For requirement (2), note that the output length of g_i is long (possibly much longer than the secret key), since it includes all the queried signatures (including the tampered signatures). However, when the event Ext_1 occurs, all the signing queries correspond to lossy tags of the encryption scheme, and hence do not reveal information about x. In particular, we can define an (inefficient) leakage function $g'_i(\cdot)$, so that (for fixed x) $g'_i(x) \approx_s g_i(x)$ are statistically close, and the (tampered) signature portion of $g'_i(x)$ cumulatively reveals very less information about x. This function g'_i precisely corresponds to the (inefficiently) generated responses to the signature queries (using the function $\mathsf{expand}_{\mathsf{K},(s,y_i)}(\mathsf{compress}_{\mathsf{K},(s,y_i)}(\cdot),\cdot)$) and the leakage queries in the proof of Theorem 7.

As shown in the proof of Theorem 7, the entropy loss induced by g_i on x given y is due to the output corresponding to $\mathcal{F}$'s leakage queries and the cumulative loss obtained from the DMWM evaluations on x (and its tampered versions). For a uniformly random x, we also learn if $\mathsf{ver}_{\mathsf{clr\text{-}owr}}(y,x) = 1$, which can reveal up to 1 additional bit of entropy. Therefore, since $\mathcal{F}$'s leakage queries were limited to being $\lambda(\kappa)$-entropy leaky, when Ext_1 occurs, the function g_i is $(\lambda(\kappa)+1)$-entropy leaky, thereby proving condition (2). □

Acknowledgments

We would like to thank the anonymous reviewers of PKC 2019 for their useful comments and suggestions. We thank Omer Paneth for pointing out to us the connection between Unique Witness Maps (UWM) and Witness encryption (WE). The first author would like to acknowledge Pandu Rangan for his involvement during the initial discussion phase of the project.

References

1. Miklós Ajtai. The shortest vector problem in L_2 is NP-hard for randomized reductions (extended abstract). In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 10–19, 1998.
2. Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In *Annual International Cryptology Conference*, pages 36–54. Springer, 2009.
3. Joël Alwen, Stephan Krenn, Krzysztof Pietrzak, and Daniel Wichs. Learning with rounding, revisited - new reduction, properties and applications. In *Annual International Cryptology Conference*, pages 57–74, 2013.
4. Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In *International conference on the theory and applications of cryptographic techniques*, pages 719–737, 2012.
5. Mihir Bellare, Igors Stepanovs, and Brent Waters. New negative results on differing-inputs obfuscation. In *International conference on the theory and applications of cryptographic techniques*, pages 792–821. Springer, 2016.
6. Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 103–112, 1988.

7. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *Annual International Cryptology Conference*, pages 41–55, 2004.
8. Dan Boneh, Richard A DeMillo, and Richard J Lipton. On the importance of eliminating errors in cryptographic computations. *Journal of cryptology*, 14(2):101–119, 2001.
9. Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 280–300. Springer, 2013.
10. Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In *International Workshop on Public Key Cryptography*, pages 501–519. Springer, 2014.
11. Elette Boyle, Gil Segev, and Daniel Wichs. Fully leakage-resilient signatures. In *International conference on the theory and applications of cryptographic techniques*, pages 89–108. Springer, 2011.
12. 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 *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 501–510. IEEE, 2010.
13. Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In *Annual International Cryptology Conference*, pages 78–96. Springer, 2006.
14. Yu Chen, Yuyu Wang, and Hong-Sheng Zhou. Leakage-resilient cryptography from puncturable primitives and obfuscation. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 575–606. Springer, 2018.
15. Ivan Damgård, Sebastian Faust, Pratyay Mukherjee, and Daniele Venturi. Bounded tamper resilience: How to go beyond the algebraic barrier. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 140–160. Springer, 2013.
16. Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Cryptography against continuous memory attacks. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 511–520. IEEE, 2010.
17. Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Efficient public-key cryptography in the presence of key leakage. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 613–631. Springer, 2010.
18. Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *International conference on the theory and applications of cryptographic techniques*, pages 523–540. Springer, 2004.
19. Antonio Faonio, Jesper Buus Nielsen, and Daniele Venturi. Mind your coins: fully leakage-resilient signatures with graceful degradation. In *International Colloquium on Automata, Languages, and Programming*, pages 456–468. Springer, 2015.
20. Antonio Faonio, Jesper Buus Nielsen, and Daniele Venturi. Predictable arguments of knowledge. In *IACR International Workshop on Public Key Cryptography*, pages 121–150. Springer, 2017.
21. Antonio Faonio and Daniele Venturi. Efficient public-key cryptography with bounded leakage and tamper resilience. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 877–907. Springer, 2016.
22. Sebastian Faust, Eike Kiltz, Krzysztof Pietrzak, and Guy N Rothblum. Leakage-resilient signatures. In *Theory of Cryptography Conference*, pages 343–360. Springer, 2010.
23. Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 416–426, 1990.
24. Eiichiro Fujisaki and Keita Xagawa. Public-key cryptosystems resilient to continuous tampering and leakage of arbitrary functions. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 908–938. Springer, 2016.
25. Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 467–476, 2013.
26. Rosario Gennaro, Anna Lysyanskaya, Tal Malkin, Silvio Micali, and Tal Rabin. Algorithmic tamper-proof (atp) security: Theoretical foundations for security against hardware tampering. In *Theory of Cryptography Conference*, pages 258–277. Springer, 2004.
27. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. On the cryptographic applications of random functions. In *Annual International Cryptology Conference*, pages 276–288, 1984.

28. Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 25–32, 1989.
29. Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Robustness of the learning with errors assumption. In *Innovations in Computer Science - ICS 2010, Tsinghua University, Beijing, China, January 5-7, 2010. Proceedings*, pages 230–240, 2010.
30. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 291–304. ACM, 1985.
31. Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.
32. Vipul Goyal, Aayush Jain, and Dakshita Khurana. Non-malleable multi-prover interactive proofs and witness signatures. Technical report, Cryptology ePrint Archive, Report 2015/1095.(2015)., 2015.
33. Susan Hohenberger, Venkata Koppula, and Brent Waters. Adaptively secure puncturable pseudorandom functions in the standard model. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 79–102. Springer, 2015.
34. Yael Tauman Kalai, Bhavana Kanukurthi, and Amit Sahai. Cryptography with tamperable and leaky memory. In *Annual International Cryptology Conference*, pages 373–390. Springer, 2011.
35. Jonathan Katz and Vinod Vaikuntanathan. Signature schemes with bounded leakage resilience. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 703–720. Springer, 2009.
36. Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 669–684. ACM, 2013.
37. Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Annual International Cryptology Conference*, pages 388–397. Springer, 1999.
38. Paul C Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *Annual International Cryptology Conference*, pages 104–113. Springer, 1996.
39. Tal Malkin, Isamu Teranishi, Yevgeniy Vahlis, and Moti Yung. Signatures resilient to continual leakage on memory and computation. In *Theory of Cryptography Conference*, pages 89–106. Springer, 2011.
40. Ralph C. Merkle. A digital signature based on a conventional encryption function. In *Annual International Cryptology Conference*, pages 369–378, 1987.
41. Silvio Micali and Leonid Reyzin. Physically observable cryptography. In *Theory of Cryptography Conference*, pages 278–296. Springer, 2004.
42. Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *International conference on the theory and applications of cryptographic techniques*, pages 700–718, 2012.
43. Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. In *Annual International Cryptology Conference*, pages 18–35, 2009.
44. Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 33–43. ACM, 1989.
45. Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 187–196, 2008.
46. Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. *SIAM Journal on Computing*, 40(6):1803–1844, 2011.
47. Gilles Piret and Jean-Jacques Quisquater. A differential fault attack technique against spn structures, with application to the aes and khazad. In *nternational workshop on cryptographic hardware and embedded systems*, pages 77–88. Springer, 2003.
48. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 84–93, 2005.
49. Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 475–484. ACM, 2014.

50. Tsz Hon Yuen, Siu Ming Yiu, and Lucas CK Hui. Fully leakage-resilient signatures with auxiliary inputs. In *Australasian Conference on Information Security and Privacy*, pages 294–307. Springer, 2012.

APPENDIX

A A Construction of C-LTDF based on the LWE Assumption.

We now construct a "relaxed" form of C-LTDFs based on the LWE assumption. Later, we argue that the relaxed lossy mode suffices for the applications in this work. Essentially, we cannot guarantee that for all pp, lossy ek and x we have that $f_{\mathsf{ek}}(x)$ is contained in some small set R_{pp}. Instead we only guarantee that this happens with overwhelming probability. That is, we relax the notion of lossy mode to the following:

- **Relaxed Lossy mode.** There exists a negligible function ν and two (inefficient) functions $\mathsf{compress}_{\mathsf{pp}} : \{0,1\}^{\ell(\kappa)} \to R_{\mathsf{pp}}$ with range $|R_{\mathsf{pp}}| \leq 2^{\ell(\kappa)-\alpha(\kappa)}$, and $\mathsf{expand}_{\mathsf{ek}}(\cdot)$ such that the following holds.

$$\Pr\left[\mathsf{eval}(\mathsf{ek},x) = \mathsf{expand}_{\mathsf{ek}}(\mathsf{compress}_{\mathsf{pp}}(x)) \;:\; \begin{array}{c}\mathsf{pp} \leftarrow \mathsf{setup}(\kappa),\\ \mathsf{ek} \leftarrow \mathsf{sample}_{\mathsf{loss}}(\kappa,\mathsf{pp}),\\ x \leftarrow \{0,1\}^{\ell(\kappa)}\end{array}\right] \geq 1-\nu(\kappa)$$

Our construction is inspired by the construction of lossy trapdoor functions from LWE of [3]. This in turn relies on the notion of learning with rounding (LWR) from [4] and a "lossy mode" technique for LWE from [29].

Definition 18 (LWE Assumption [48]). *Let κ be the security parameter, $n = n(\kappa), m = m(\kappa), q = q(\kappa)$ be integers and let $\chi = \chi(\kappa)$ be a distribution over $\mathbb{Z}_q$. The $\mathrm{LWE}_{n,m,q,\chi}$ assumption says that for $\mathbf{A} \leftarrow \mathbb{Z}_q^{m\times n}$, $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow \chi^m$, the distribution $(\mathbf{A}, \mathbf{s}\cdot\mathbf{A}+\mathbf{e})$ is computationally indisitinguishable from $(\mathbf{A},\mathbf{u})$ where $\mathbf{u} \leftarrow \mathbb{Z}_q^m$.*

Let $p < q$ be integers. We define the rounding function

$$\lfloor\cdot\rceil_p \;:\; \mathbb{Z}_q \to \mathbb{Z}_p \;:\; x \mapsto \lfloor (p/q)\cdot x\rceil,$$

For a vector or matrix $\mathbf{A}$, we let $\lfloor\mathbf{A}\rceil_p$ denote the output of applying the function $\lfloor\cdot\rceil_p$ on each component.

For any integer $\tau > 0$ we define the set of elements that are within distance τ of the border between two intervals that round to different values:

$$\mathsf{border}_{p,q}(\tau) = \{x \in \mathbb{Z}_q \;:\; \exists y \in \mathbb{Z}, |y| \leq \tau, \lfloor x\rceil_p \neq \lfloor x+y\rceil_p\}\,.$$

Note that we can test membership in this set efficiently. Furthermore it's easy to see that:

Lemma 7 ([3]). *For every p, q, τ it holds that $\Pr_{x\leftarrow\mathbb{Z}_q}[x \in \mathsf{border}_{p,q}(\tau)] \leq \frac{2\tau p}{q}$.*

Using well known results on trapdoors for LWE [1, 42] we also get the following trapdoor for learning with rounding (LWR).

Lemma 8 (Trapdoors for LWR [3]). *For $n \geq 1$, $q \geq 2$, there exist sufficiently large polynomials sufficiently large $m = O(n \log q)$ and $p = O(\sqrt{mn \log q})$, there exist algorithms* $(\mathsf{GenTrap}, \mathsf{LWRInvert})$ *such that:*

- $(\mathbf{A}, \mathsf{tk}) \leftarrow \mathsf{GenTrap}(1^n, 1^m, q)$ *outputs a matrix* $\mathbf{A}$ *which is statistically close to uniform over* $\mathbb{Z}_q^{n \times m}$.
- *For any* $(\mathbf{A}, \mathsf{tk})$ *in the support of* $\mathsf{GenTrap}(1^n, 1^m, q)$ *and any* $\mathbf{s} \in \{0,1\}^n$ *we have* $\mathsf{LWRInvert}_{\mathsf{tk}}(\lfloor \mathbf{s} \cdot \mathbf{A} \rfloor_p) = \mathbf{s}$.

We are now ready to give a construction of C-LTDFs from LWE. Let $n = n(\kappa), m = m(\kappa), n' = n'(\kappa), q = q(\kappa), p = p(\kappa)$ be integers and let $\chi = \chi(\kappa)$ be a distribution over $\mathbb{Z}$. We require that the support of χ lies in an interval $[-v, v]$ such that $v/q = \mathsf{negl}(\kappa)$. We further require that n, m, q, p satisfy the requirements of Lemma 8.

1. $\mathsf{setup}(\kappa)$: Choose $\mathbf{B} \leftarrow \mathbb{Z}_q^{n \times n'}$, $\mathbf{C} \leftarrow \mathbb{Z}_q^{n' \times m}$ and $E \leftarrow \chi^{n \times m}$. Let $\mathbf{P} = \mathbf{BC} + \mathbf{E} \in \mathbb{Z}_q^{n \times m}$. Output $\mathsf{pp} = \mathbf{P}$.
2. $\mathsf{sample}_{\mathsf{inj}}(\kappa, \mathsf{pp})$: Choose $(\mathbf{A}, \mathsf{tk}) \leftarrow \mathsf{GenTrap}(1^n, 1^m, q)$ and sett $\mathsf{ek} = \mathbf{A}$. Output $(\mathsf{ek}, \mathsf{tk})$.
3. $\mathsf{sample}_{\mathsf{loss}}(\kappa, \mathsf{pp})$: Parse $\mathsf{pp} = \mathbf{P} \in \mathbb{Z}_q^{n \times m}$. Choose $\mathbf{R} \leftarrow \{0,1\}^{m \times m}$ and set $\mathbf{A} = \mathbf{P} \cdot \mathbf{R}$. Output $\mathsf{ek} = \mathbf{A}$.
4. $\mathsf{eval}(\mathsf{ek}, \boldsymbol{x})$: Parse $\mathsf{ek} = \mathbf{A} \in \mathbb{Z}_q^{n \times m}$. On input vector $\boldsymbol{x} \in \{0,1\}^n$ output $\lfloor \boldsymbol{x} \cdot \mathbf{A} \rfloor_p$.
5. $\mathsf{invert}(\mathsf{ek}, \mathsf{tk}, \boldsymbol{y})$: Output $\mathsf{LWRInvert}_{\mathsf{tk}}(\boldsymbol{y})$.

Theorem 11. *Under the* $\mathrm{LWE}_{n',m,q,\chi}$ *assumption, the above is an* $(n, n' \log q)$-C-LTDF *with "relaxed" lossiness and error* $\nu(\kappa) = 2nm^2vp/q = \mathsf{negl}(\kappa)$.

Proof. First, let us show the indistinguishability of lossy and injective modes. In injective mode, we have $\mathsf{ek} = \mathbf{A}$ is statistically close to uniform by Lemma 8. In lossy mode, we have $\mathsf{ek} = \mathbf{A} = \mathbf{P} \cdot \mathbf{R} = (\mathbf{BC} + \mathbf{E})\mathbf{R}$. We first apply the LWE assumption to replace $\mathbf{P}$ by uniform (here we think of the rows of $\mathbf{B}$ as the LWE secrets and the matrix $\mathbf{C}$ as the LWE coefficients). Next we apply the leftover hash lemma to argue that $\mathbf{A} = \mathbf{P} \cdot \mathbf{R}$ is statistically close to uniform. Therefore, in both modes, ek is computationally indistinguishable from uniform and therefore the modes are indistinguishable from each other.

Secondly, in injective mode, the inversion algorithm is correct by Lemma 8.

Thirdly, let $\mathsf{pp} = \mathbf{P} = \mathbf{BC} + \mathbf{E}$ and let $\mathsf{ek} = \mathbf{A} = \mathbf{P} \cdot \mathbf{R}$ for some $\mathbf{R} \in \{0,1\}^{m \times m}$. Let $\boldsymbol{x} \in \{0,1\}^n$. Then $\mathsf{eval}(\mathsf{ek}, \boldsymbol{x}) = \lfloor \boldsymbol{x}\mathbf{A} \rfloor_p = \lfloor \boldsymbol{x}(\mathbf{BC} + \mathbf{E})\mathbf{R} \rfloor_p$. Furthermore $\lfloor \boldsymbol{x}(\mathbf{BC} + \mathbf{E})\mathbf{R} \rfloor_p = \lfloor \boldsymbol{x}(\mathbf{BCR}) \rfloor_p$ unless one of the components of $\boldsymbol{x}\mathbf{BCR}$ lies in $\mathsf{border}_{p,q}(nmv)$, which (by Lemma 7 and union bound over the m components) happens with probability at most $\nu(\kappa) = 2nm^2vp/q = \mathsf{negl}(\kappa)$. Therefore if we define $\mathsf{compress}_{\mathsf{pp}}(\boldsymbol{x}) = \boldsymbol{x}\mathbf{B}$ and $\mathsf{expand}_{\mathsf{ek}}(\boldsymbol{z}) = \lfloor \boldsymbol{z}\mathbf{CR} \rfloor_p$ then $\Pr[\mathsf{eval}(\mathsf{ek}, \boldsymbol{x}) = \mathsf{expand}_{\mathsf{ek}}(\mathsf{compress}_{\mathsf{pp}}(\boldsymbol{x}))] \geq 1 - \nu(\kappa)$.

A.1 Relying on Relaxed Lossy Mode

We now discuss how the rest of the paper is affected by using a C-LTDF with relaxed lossiness. In particular, we sketch the changes that need to be made for all of the results to go through as previously by starting with relaxed lossiness.

From C-LTDF to C-ALBO-LTDF. We can define a relaxed lossy mode for C-ALBO-LTDF analogously to that of C-LTDF. That is we now require that:

- **Relaxed Lossy mode.** There exists a negligible function ν and two (inefficient) functions $\mathsf{compress}_{\mathsf{ek}} : \{0,1\}^{\ell(\kappa)} \to R_{\mathsf{ek}}$ with range $|R_{\mathsf{ek}}| \leq 2^{\ell(\kappa)-\alpha(\kappa)}$, and $\mathsf{expand}_{\mathsf{ek},b}(\cdot)$ such that the following holds. For all $b^* \in \mathcal{B}$,

$$\Pr\left[\forall\ b \in \mathcal{B} \setminus \{b^*\}\ \ \mathsf{eval}(\mathsf{ek}, b, x) = \mathsf{expand}_{\mathsf{ek},b}(\mathsf{compress}_{\mathsf{ek}}(x))\right] \geq 1 - \nu(\kappa)$$

  where the probability is over $\mathsf{ek} \leftarrow \mathsf{sample}_{\mathsf{c\text{-}albo}}(\kappa, b^*), x \leftarrow \{0,1\}^{\ell(\kappa)}$.

We would like to say that the construction of C-ALBO-TDF from C-LTDF and iO in Section 4.2.3 also works for the relaxed notion of lossiness. Unfortunately, we cannot do so directly. In that construction we are sampling the lossy keys $\widetilde{\mathsf{ek}}$ of the C-LTDF pseudorandomly inside of an obfuscated program by applying a PRF to the branch index b. We therefore cannot directly rely on the relaxed lossy mode property of the C-LTDF which requires that the lossy key $\widetilde{\mathsf{ek}}$ is chosen randomly. To overcome this issue, we modify the construction of C-ALBO-TDF, and in particular, the obfuscated program as follows. We now hard-code a uniformly random pad into the obfuscated program and, in the "else" part, we now use the randomness $r_{\mathsf{loss}} = F(K, b) \oplus \mathsf{pad}$ to sample the lossy key $(\widetilde{\mathsf{ek}}, \bot) \leftarrow \mathsf{sample}_{\mathsf{loss}}(\kappa, \mathsf{pp}; r_{\mathsf{loss}})$. This ensures that for any particular branch $b \in \mathcal{B} \setminus \{b^*\}$ fixed a-priori we have $\Pr\left[\mathsf{eval}(\mathsf{ek}, b, x) \neq \mathsf{expand}_{\mathsf{ek},b}(\mathsf{compress}_{\mathsf{ek}}(x))\right] \leq \nu(\kappa)$ where ν is the lossiness error of the underlying C-LTDF. We can then apply a union bound over all $b \in \mathcal{B} \setminus \{b^*\}$ to argue that we get the relaxed lossiness for the C-ALBO-TDF with error $|\mathcal{B}| \cdot \nu(\kappa)$. By ensuring that the error ν of unrelying C-LTDF is small enough, we can make this negligible. Luckily, the LWE construction allows us to set the error to be an arbitarily small $2^{-\mathsf{poly}(\kappa)}$.

From C-ALBO-TDF to DMWM. Lastly, we need to ensure that a C-ALBO-TDF with relaxed lossiness is good enough for the construction of DMWM. This holds because the lossiness is used inside of an information theoretic argument, which is itself used to argue lossiness of DMWM (see Section 5.2). In particular, we can switch from computing $\mathsf{eval}_{\mathsf{c\text{-}albo}}(\mathsf{ek}, \mathsf{tag}', w)$ to $\mathsf{expand}_{\mathsf{ek},\mathsf{tag}'}(\mathsf{compress}_{\mathsf{ek}}(w))$ for each $\mathsf{tag}' \neq \mathsf{tag}$ and this only induces a negligible difference in probabilities. The rest of the argument proceeds as previously.