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 $1 - o(1)$.

1 Introduction

A foundational innovation of theoretical computer science has been the generalization of the notion of what a *proof* is. Interactive proofs, zero-

[⋆] 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` 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 [26], 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 [4] 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 [21] 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 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

[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 $x \notin L$.

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 $(\mathsf{setup}, \mathsf{map}, \mathsf{check})$. 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 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 [41] 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 [28], building on the notion of "Signatures of Knowledge" [12]; 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. [3], who relied on *sub-exponentially secure* Indistin-

[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.

guishability 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) [23]. 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) [38], 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 [38] 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 [31,32]) and tampering (see for instance [6, 40]). 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 [9, 14, 19, 29, 33]. 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 [9, 13, 17], and not against tampering attacks (as explicitly posed in [19]). 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 [13], 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. [14]. 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 [14,19], and motivates the need for finding a deterministic solution. A recent work of Chen et al. [13] 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 [9] and tamper-resilience [19], 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 model, 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 [1] and Katz and Vaikuntanathan [29] 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 [1] was in the random oracle (RO) model. [29] 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 [20] gave a construction of a *stateful* LRS scheme in the "Only Computation Leaks" model of Micali and Reyzin [35]. 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. [9] and Malkin et al. [33]. Faonio et al. [17] 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. [42] 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. [13] gave an FLR signature scheme with a deterministic signing algorithm, and achieved selective unforgeability, relying on $i\mathcal{O}$.

Tamper resilience was addressed in [14,19]. 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 [19]. The continual memory leakage (CML) model has been studied in [10,15,33].

Comparison with the work of [13]. Recently, Chen et al. [13] 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 [18]. Predictable argument of knowledge (PAoK) [18] 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 [41]. 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 $\textsc{uwm}.\mathsf{map}(\mathsf{K}, \mathsf{stmnt}, w)$ 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^*

[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.

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 [41] implicitly constructs one using iO as a part of their construction of NIZKs, and Bellare et al. [3] 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 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 [7,8,24,30]. 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) [23]. 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 and 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 [15] 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 $f(b, \dot{})$ 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 [38]. 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 [2], which relies on the "lossy mode" of LWE from [25].

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 Puncturable Digital Signature Schemes

A puncturable digital signature (PDS) scheme [3] 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. [3], 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^*. A formal definition is provided in the full version of our paper [11].

Below, we summarize the construction of our PDS scheme, and refer to the full version [11] for the formal details of the scheme. Our construction of the PDS relies on the *sole* assumption that one-way functions exist.

The construction follows the paradigm of extending one-time signatures into full-fledged signatures using a tree of pseudorandomly generated key pairs [27,34,37]. 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.

3 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 1 (Compact Witness Map (CWM)). *For $\alpha \geq 0$, an α-CWM for an* **NP** *relation R is a triple* $\textsc{cwm} = (\mathsf{setup}, \mathsf{map}, \mathsf{check})$ *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 $\mathcal{A}$,* $\mathsf{Adv}_{\mathcal{A}}^{\mathrm{CWM}}(\kappa)$ *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 $\textsc{cwm}.\mathsf{setup}$ 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 3.1).

Definition 2 (Dual Mode Witness Maps (DMWM)). *An α-DMWM with tag space $\mathcal{T}$ for an* **NP** *relation R is a tuple* $\textsc{dmwm} = (\mathsf{setup}, \mathsf{map}, \mathsf{check}, \mathsf{extract})$ *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* $\mathcal{A}$*,* $\mathsf{Adv}_{\mathcal{A}}^{\text{DMWM-hide}}(\kappa)$ *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* $\mathcal{A}$*,* $\mathsf{Adv}_{\mathcal{A}}^{\text{DMWM}}(\kappa)$ *defined below is negligible:*

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

- *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* $|S_{\mathsf{K},x}| \le 2^{\alpha(\kappa)}$*, and for all* $\mathsf{tag}' \neq \mathsf{tag}$*,* $\mathsf{map}(\mathsf{K}, \mathsf{tag}', x, w) = \mathsf{expand}_{\mathsf{K},x}(\mathsf{compress}_{\mathsf{K},x}(w), \mathsf{tag}')$*.*

3.1 Unique Witness Maps

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

3.1.1 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 $i\mathcal{O}$).

Theorem 1. *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.*

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 $P \leftarrow i\mathcal{O}(\mathsf{Endorse}_{\mathsf{sk}}^{R_\ell})$, where the program $\mathsf{Endorse}_{\mathsf{sk}}^{R_\ell}$ 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 $\mathsf{Endorse}_{\mathsf{sk}}^{R_\ell}((x, w))$
    Constant: Signing key $\mathsf{sk}$
    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 $\mathsf{pEndorse}_{\widehat{\mathsf{sk}}_{x^*}}^{R_\ell}((x, w))$
    Constant:
    $\boxed{\text{Punctured signing key } \widehat{\mathsf{sk}}_{x^*}}$
    Input Domain: $(x, w) \in \{0,1\}^\ell \times \{0,1\}^{\ell'}$
    if $(x, w) \in R_\ell$ and $\boxed{x \neq x^*}$ then
        output $\boxed{\text{PDS}.\mathsf{psign}(\widehat{\mathsf{sk}}_{x^*}, x)}$
    else
        output $\perp$
```

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

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 $\mathsf{Adv}_{\mathcal{A}}^{\text{UWM}}(\kappa)$. 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 $\mathsf{Endorse}_{\mathsf{sk}}^{R_\ell}$, the program $\mathsf{pEndorse}_{\widehat{\mathsf{sk}}_{x^*}}^{R_\ell}$ (also shown in Figure 1) is used, where $\widehat{\mathsf{sk}}_{x^*} \leftarrow$ 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 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 $\mathsf{Endorse}_{\mathsf{sk}}^{R_\ell}$ and $\mathsf{pEndorse}_{\widehat{\mathsf{sk}}_{x^*}}^{R_\ell}$ 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 $\textsc{pds}.\mathsf{sign}(\mathsf{sk}, x) = \textsc{pds}.\mathsf{sign}(\widehat{\mathsf{sk}}_{x^*}, x)$. So $\mathcal{A}_{i\mathcal{O}}$ can output the pair of programs $\mathsf{Endorse}_{\mathsf{sk}}^{R_\ell}$ and $\mathsf{pEndorse}_{\widehat{\mathsf{sk}}_{x^*}}^{R_\ell}$. It receives back an obfuscated program P and carries out the rest of the UWM security game with $\mathcal{A}$ using P. If $P \leftarrow i\mathcal{O}(\mathsf{Endorse}_{\mathsf{sk}}^{R_\ell})$, 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 (vk, P), where $(\mathsf{vk}, \mathsf{sk}) \leftarrow \textsc{pds}.\mathsf{keygen}(\ell, \kappa)$, and P is generated from the punctured secret-key $\widehat{\mathsf{sk}}_{x^*}$, outputs a purported signature w^*, and wins if $\textsc{pds}.\mathsf{ver}(\mathsf{vk}, x^*, w^*) = 1$. By the security of $\textsc{pds}$, the probability of $\mathcal{A}$ winning is at most $\mathsf{Adv}_{\mathcal{A}}^{\textsc{pds}}(\kappa)$, which is negligible. $\square$

Remark 1. In the above proof, we only show selective soundness of $\textsc{uwm}$. 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 $\textsc{pds}$ 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 $\mathsf{Adv}_{\mathcal{A}}^{\textsc{pds}}(\kappa_1) \leq 2^{-(\ell+\kappa)}$ and $\mathsf{Adv}_{Samp,\mathcal{D}}^{i\mathcal{O}}(\kappa_2) \leq 2^{-(\ell+\kappa)}$, where κ_1 and κ_2 are the security parameters for $\textsc{pds}$ and $i\mathcal{O}$ respectively, and κ is the security parameter for $\textsc{uwm}$). One can set κ_1 and κ_2 to be large enough to satisfy this.

3.1.2 Implication to Witness Encryption.

In this section, we show that UWM implies Witness encryption (WE). Due to space constraints, we only present a high level idea behind the construction and refer the reader to our full version [11] for the detailed description.

Intuition behind the construction. 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. 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 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.*

3.2 New Kinds of Lossy Trapdoor Functions

3.2.1 Cumulative Lossy Trapdoor Functions. Here 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}. Then, mapping a random x to $f(x)$ loses 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. The formal definition follows.

Definition 3 (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.*

3.2.1.1 **C-LTDF from the d-Linear Assumption.** Due to space constraints, we present the construction of C-LTDF from the d-linear assumption, and refer the reader to our full version [11] for the construction from LWE.

The d-linear assumption [5] is a generalization of the Decision Diffie-Hellman (DDH) assumption. 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 [36]. Due to space constraints, we only specify the d-Linear assumption here, and refer the reader to our full version [11] for the definition of Matrix d-Linear assumption.

Definition 4 (d-Linear assumption [5]). *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:*

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

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

Before specifying the assumption, we will need 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.

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 $\boldsymbol{x} \in \{0,1\}^n$, 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 2. *Suppose the d-Linear assumption holds for* $\mathsf{GroupGen}$*. 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 $\epsilon > d \log_2 p_{\max}(\kappa)/n(\kappa)$.*

Due to space constraints, we present the proof in our full version [11].

3.2.2 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 [39]. 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 5 (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 lossy 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 $b^* \in \mathcal{B}$,* $\mathsf{sample}_{\text{c-albo}}(\kappa, b^*)$ *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^*,* $\mathsf{invert}_{\text{c-albo}}(\mathsf{tk}, b^*, \mathsf{eval}_{\text{c-albo}}(\mathsf{ek}, b^*, x)) = x$ *for all $x \in \{0,1\}^{\ell(\kappa)}$ (i.e.,* $\mathsf{eval}_{\text{c-albo}}(\mathsf{ek}, b^*, \cdot)$ *computes an injective function $g_{\mathsf{ek},b^*}(\cdot)$ over the domain $\{0,1\}^{\ell(\kappa)}$, and* $\mathsf{invert}_{\text{c-albo}}(\mathsf{tk}, b^*, \cdot)$ *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.*

3.2.3 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. Due to space constraints, we refer to the full version of our paper [11] for the detailed construction.

Theorem 3. *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 *sketched above is a collection of* (ℓ, α)*-cumulative all-lossy-but-one trapdoor functions.*

The detailed proof of this theorem is given in full version [11].

3.3 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 2). 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 well 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 3) for the following language:

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

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. 2. Construction of DMWM for an **NP** relation R_ℓ.

Theorem 4. *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 2 is α''-DMWM with tag space $\mathcal{T} = \mathcal{B}$ for the* **NP** *relation R_ℓ.*

The detailed proof of this theorem is given in full version of our paper [11].

4 Fully Leakage and Tamper-resilient Signature Scheme

A signature scheme with setup $\textsc{sig}$ is a tuple of PPT algorithms $\textsc{sig} = (\mathsf{setup}, \mathsf{keygen}, \mathsf{sign}, \mathsf{verify})$. 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 $\mathsf{pub} \leftarrow \textsc{sig}.\mathsf{setup}(\kappa)$, any key pair $(\mathsf{ssk}, \mathsf{vk})$ produced by $\textsc{sig}.\mathsf{keygen}$ and all messages $m \in \mathcal{M}$, we require $\textsc{sig}.\mathsf{verify}(\mathsf{vk}, (m, \textsc{sig}.\mathsf{sign}(\mathsf{ssk}, m))) = 1$.

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 $\textsc{sig}.\mathsf{sign}$ 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 $\textsc{sig}.\mathsf{sign}(\widetilde{\mathsf{ssk}}, \cdot, \widetilde{r})$, 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.

4.1 Security model for FLTR signatures.

Definition 6. **((λ, t)-FLTR security).** *We say that a signature scheme* $\textsc{sig} = (\textsc{sig}.\mathsf{setup}, \textsc{sig}.\mathsf{keygen}, \textsc{sig}.\mathsf{sign}, \textsc{sig}.\mathsf{verify})$ *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* $\Pr\left[\mathsf{Success}_{\Pi,\mathcal{F}}^{(\lambda,t)\text{-}\mathsf{FLTR}}(\kappa)\right] \leq \mathsf{negl}(\kappa)$*, where the event* $\mathsf{Success}_{\Pi,\mathcal{F}}^{(\lambda,t)\text{-}\mathsf{FLTR}}(\kappa)$ *is defined via the following experiment between a challenger* $\mathcal{C}$ *and the forger* $\mathcal{F}$*:*

1. *Initially, the challenger* $\mathcal{C}$ *computes* $\mathsf{pub} \leftarrow \textsc{sig}.\mathsf{setup}(\kappa)$ *and* $(\mathsf{ssk}, \mathsf{vk}) \leftarrow \textsc{sig}.\mathsf{keygen}(\kappa, \mathsf{pub})$*, 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* $\textsc{sig}.\mathsf{sign}^*_{\mathsf{ssk}}(\cdot)$ *receives as input a message* $m_i \in \mathcal{M}$*. The challenger* $\mathcal{C}$ *then samples* $r_i \leftarrow \mathcal{R}$*, and computes* $\sigma_i \leftarrow \textsc{sig}.\mathsf{sign}(\mathsf{ssk}, m, r_i)$*. 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* $\textsc{sig}.\mathsf{sign}(\widetilde{\mathsf{ssk}}_i, \cdot, \widetilde{r}_i)$ *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.*

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

- *The signature* σ^* *verifies with respect to the original verification key* vk*, i.e.,* $\textsc{sig}.\mathsf{verify}(\mathsf{vk}, (m^*, \sigma^*)) = 1$.
- 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 4.2) 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.

4.2 Construction of our FLTR signature scheme.

In this section, we present our construction of FLTR signature scheme. In Figure 3, we present this construction.

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

$$L := \big\{ (s, y) \ : \ \exists\ x \ \mathsf{s.t.}\ y = \textsc{spr}.\mathsf{eval}_s(x)) \ \big\}$$

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

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

Fig. 3. Construction of FLTR Signature Scheme $\textsc{sig}$

Theorem 5. *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* $\rho(\kappa) = \frac{t(\kappa)}{d(\kappa)} = O(1/\kappa)$*.*

Due to space constraints, we present the proof of Theorem 5 in the full version of our paper [11].

Extension to Continuous Leakage and Tampering. Our construction can be readily extended to a model of continuous leakage and tampering, with periodic (tamper-proof) key updates. To this end, first we note that we can replace the SPR function family used in our construction with a 'entropy-bounded" or "noisy" leakage-resilient one-way relations (LR-OWR) [9, 16]. Then, we show that the only modification required to upgrade our LTR signature construction to the setting of continuous leakage and tampering is to further replace the noisy LR-OWR above with its continuous leakage analogue, which we call noisy *continuous* LR-OWR (CLR-OWR), as defined by Dodis et al. [15]. Our construction bypasses the impossibility result of [22] by allowing the signing key to periodically update in between leakage and tampering queries. We refer the reader to the full version [11] for further details.

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. Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In *CRYPTO*, pages 36–54. Springer, 2009.
2. Joël Alwen, Stephan Krenn, Krzysztof Pietrzak, and Daniel Wichs. Learning with rounding, revisited - new reduction, properties and applications. In *CRYPTO*, pages 57–74, 2013.
3. Mihir Bellare, Igors Stepanovs, and Brent Waters. New negative results on differing-inputs obfuscation. In *EUROCRYPT*, pages 792–821. Springer, 2016.

4. Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *STOC*, pages 103–112, 1988.
5. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *CRYPTO*, pages 41–55, 2004.
6. 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.
7. Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In *ASIACRYPT*, pages 280–300. Springer, 2013.
8. Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In *PKC*, pages 501–519. Springer, 2014.
9. Elette Boyle, Gil Segev, and Daniel Wichs. Fully leakage-resilient signatures. In *EUROCRYPT*, pages 89–108. Springer, 2011.
10. Zvika Brakerski, Yael Tauman Kalai, Jonathan Katz, and Vinod Vaikuntanathan. Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In *FOCS*, pages 501–510. IEEE, 2010.
11. Suvradip Chakraborty, Manoj Prabhakaran, and Daniel Wichs. Witness maps and applications. Cryptology ePrint Archive, Report 2020/090, 2020. https://eprint.iacr.org/2020/090.
12. Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In *CRYPTO*, pages 78–96. Springer, 2006.
13. Yu Chen, Yuyu Wang, and Hong-Sheng Zhou. Leakage-resilient cryptography from puncturable primitives and obfuscation. In *ASIACRYPT*, pages 575–606. Springer, 2018.
14. Ivan Damgård, Sebastian Faust, Pratyay Mukherjee, and Daniele Venturi. Bounded tamper resilience: How to go beyond the algebraic barrier. In *ASIACRYPT*, pages 140–160. Springer, 2013.
15. Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Cryptography against continuous memory attacks. In *FOCS*, pages 511–520. IEEE, 2010.
16. Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Efficient public-key cryptography in the presence of key leakage. In *ASIACRYPT*, pages 613–631. Springer, 2010.
17. Antonio Faonio, Jesper Buus Nielsen, and Daniele Venturi. Mind your coins: fully leakage-resilient signatures with graceful degradation. In *ICALP*, pages 456–468. Springer, 2015.
18. Antonio Faonio, Jesper Buus Nielsen, and Daniele Venturi. Predictable arguments of knowledge. In *PKC*, pages 121–150. Springer, 2017.
19. Antonio Faonio and Daniele Venturi. Efficient public-key cryptography with bounded leakage and tamper resilience. In *ASIACRYPT*, pages 877–907. Springer, 2016.
20. Sebastian Faust, Eike Kiltz, Krzysztof Pietrzak, and Guy N Rothblum. Leakage-resilient signatures. In *TCC*, pages 343–360. Springer, 2010.
21. Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *STOC*, pages 416–426, 1990.
22. Eiichiro Fujisaki and Keita Xagawa. Public-key cryptosystems resilient to continuous tampering and leakage of arbitrary functions. In *ASIACRYPT*, pages 908–938. Springer, 2016.
23. Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In *STOC*, pages 467–476, 2013.

24. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. On the cryptographic applications of random functions. In *CRYPTO*, pages 276–288, 1984.
25. 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.
26. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems. In *STOC*, pages 291–304. ACM, 1985.
27. Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, April 1988.
28. 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.
29. Jonathan Katz and Vinod Vaikuntanathan. Signature schemes with bounded leakage resilience. In *ASIACRYPT*, pages 703–720. Springer, 2009.
30. Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In *ACM CCS*, pages 669–684. ACM, 2013.
31. Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *CRYPTO*, pages 388–397. Springer, 1999.
32. Paul C Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *CRYPTO*, pages 104–113. Springer, 1996.
33. Tal Malkin, Isamu Teranishi, Yevgeniy Vahlis, and Moti Yung. Signatures resilient to continual leakage on memory and computation. In *TCC*, pages 89–106. Springer, 2011.
34. Ralph C. Merkle. A digital signature based on a conventional encryption function. In *CRYPTO*, pages 369–378, 1987.
35. Silvio Micali and Leonid Reyzin. Physically observable cryptography. In *TCC*, pages 278–296. Springer, 2004.
36. Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. In *CRYPTO*, pages 18–35, 2009.
37. Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *STOC*, pages 33–43. ACM, 1989.
38. Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In *STOC*, pages 187–196, 2008.
39. Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. *SIAM Journal on Computing*, 40(6):1803–1844, 2011.
40. Gilles Piret and Jean-Jacques Quisquater. A differential fault attack technique against spn structures, with application to the aes and khazad. In *CHES*, pages 77–88. Springer, 2003.
41. Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *STOC*, pages 475–484. ACM, 2014.
42. Tsz Hon Yuen, Siu Ming Yiu, and Lucas CK Hui. Fully leakage-resilient signatures with auxiliary inputs. In *ACISP*, pages 294–307. Springer, 2012.