

A Map of Witness Maps: New Definitions and Connections

Suvradip Chakraborty^{1*}, Manoj Prabhakaran^{2**}, and Daniel Wichs^{3***}

¹ Visa Research

`suvradip1111@gmail.com`

² Department of Computer Science, IIT Bombay

`mp@cse.iitb.ac.in`

³ Northeastern University and NTT Research

`wichs@ccs.neu.edu`

Abstract. A *witness map* deterministically maps a witness w of some NP statement x into computationally sound proof that x is true, with respect to a public common reference string (CRS). In other words, it is a deterministic, non-interactive, computationally sound proof system in the CRS model. A *unique witness map* (UWM) ensures that for any fixed statement x , the witness map should output the same *unique* proof for x , no matter what witness w it is applied to. More generally a *compact witness map* (CWM) can only output one of at most 2^α proofs for any given statement x , where α is some compactness parameter. Such compact/unique witness maps were proposed recently by Chakraborty, Prabhakaran and Wichs (PKC '20) as a tool for building tamper-resilient signatures, who showed how to construct UWMs from indistinguishability obfuscation (iO). In this work, we study CWMs and UWMs as primitives of independent interest and present a number of interesting connections to various notions in cryptography.

- First, we show that UWMs lie somewhere between witness PRFs (Zhandry; TCC '16) and iO – they imply the former and are implied by the latter. In particular, we show that a relaxation of UWMs to the “designated verifier (dv-UWM)” setting is *equivalent* to witness PRFs. Moreover, we consider two flavors of such dv-UWMs, which correspond to two flavors of witness PRFs previously considered in the literature, and show that they are all in fact equivalent to each other in terms of feasibility.
- Next, we consider CWMs that are extremely compact, with $\alpha = O(\log \kappa)$, where κ is the security parameter. We show that such CWMs imply *pseudo-UWMs* where the witness map is allowed to be *pseudo-deterministic* – i.e., for every true statement x , there is a unique proof such that, on any witness w , the witness map outputs this proof with $1 - 1/p(\lambda)$ probability, for a polynomial p that we can set arbitrarily large.

* Work done while the author was at ETH Zurich.

** Supported by the IITB Trust Lab.

*** Research supported by NSF grant CNS-1750795, CNS-2055510, the Alfred P. Sloan Research Fellowship.

- Lastly, we consider CWMs that are mildly compact, with $\alpha = p(\lambda)$ for some a-priori fixed polynomial p , independent of the length of the statement x or witness w . Such CWMs are implied by succinct non-interactive arguments (SNARGs). We show that such CWMs imply NIZKs, and therefore lie somewhere between NIZKs and SNARGs.

1 Introduction

When several mathematicians prove the same theorem, it is unlikely that they would all write down the exact same proof. Similarly, in the context of NP, a true statement (e.g., that some graph is 3-colorable) will often have many different proofs/witnesses (e.g., 3-colorings of the vertices). Can we come up with a proof system for NP languages where the proofs are guaranteed to be unique?

This question was studied extensively in complexity theory, where the class of languages with unique proofs is known as UP [15]. It is believed to be unlikely that $\text{NP} = \text{UP}$, meaning that we do not believe that all NP languages have unique proof systems, and there are several results that separate the two classes relative to oracles [1, 2, 13]. Recently, the work of Chakraborty, Prabhakaran and Wichs [4] proposed unique proof systems with *computational soundness* (aka arguments). They defined the notion of a *unique witness map* (UWM) in the common reference string (CRS) model. This is a deterministic polynomial-time map that takes as input an NP statement x and some arbitrary witness w for x (and the CRS) and maps them to a unique proof w^* for x . Any other witness w' for x is mapped to the same unique proof w^* . There is also a polynomial time verifier that checks whether w^* is a good proof of the statement x . The computational soundness guarantee ensures that no polynomial time adversary can cause the verifier to accept a proof of a false statement x , except with negligible probability over the choice of the CRS. In other words, a UWM is a deterministic non-interactive computationally sound proof (aka argument) system with unique proofs. More generally, [4] also considered a relaxation of UWMs to *compact witness maps* (CWMs) where the number of possible proofs w^* that the map can output for any given statement x is bounded by 2^α , for some *compactness parameter* α . UWMs then naturally correspond to CWMs with $\alpha = 0$.

It is worth noting that UWMs/CWMs only impose a restriction on the number of proofs w^* that the prover outputs, but not on the number of proofs that the verifier accepts for a given statement x . It may be the case that we have a UWM where the prover outputs a unique proof w^* for

a given statement x , but there are exponentially many alternate proofs that the verifier would accept as well. We also note that UWMs are easily seen to be a special case of a *witness-indistinguishable* proof system, since all witnesses are mapped to the same unique proof.

The work of [4] showed how to construct UWMs from indistinguishability obfuscation (iO) and one-way functions, closely following the construction of NIZKs due to [14]. They also showed that UWMs imply witness encryption. As their main result, they gave an application of UWMs to the problem of leakage and tamper-resilient signatures with a deterministic signer. However, not much else was known about UWMs/CWMs and how they relate to other notions in cryptography.

1.1 Our Results

In this work, we undertake the thorough study of UWMs/CWMs as primitives of interest in their own right. We provide a number of novel results to better understand these notions and discover surprising connections between UWM/CWM and other cryptographic objects of interest. Interestingly, we show that (quantitative) compression factor affects the (qualitative) cryptographic power, leading to a hierarchy of “worlds”, depending on whether all of NP has α -CWM for, say, $\alpha = 0, O(1), \log n, n^c$ ($c < 1$), somewhat akin to Impagliazzo’s worlds.

The study of UWMs/CWMs can be seen as part of a broader context of complexity theoretic study within cryptography, whose aim is to understand connections between primitives and their relative power. We also view the study of UWMs/CWMs as adding to the understanding of “functional compression” as a fundamental cryptographic feature. For example, functional compression plays a central role in obfuscation, where we can define variants (e.g., XiO vs iO [11]) depending on the level of compression provided. And (perhaps a bit further off), the complexity of computing Kolmogorov complexity, which is also about functional compression, is deeply related to the existence of one-way functions [12].

We now discuss each of our results, relating CWMs with various levels of compression to other cryptographic objects and to each other.

Relating UWMs and witness PRFs. At its most compact end, witness maps take the form of UWMs. We show several results tightly relating flavors of UWMs and flavors of witness PRFs.

First, we show that UWMs imply witness PRFs [17], which lie somewhere between witness encryption and iO, but are believed to be strictly stronger than witness encryption. In particular, they were shown to imply multi-party key exchange without trusted setup, polynomially-many

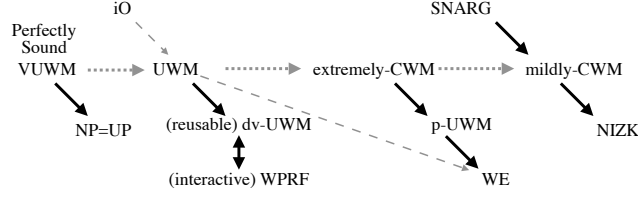


Fig. 1. A summary of the implications established (under standard cryptographic assumptions). The dotted lines correspond to trivial implications, and the dashed line are results from [4].

hardcore bits for any one-way function, and several other applications that are otherwise only known from iO , but not from witness encryption.

In a witness PRF, just like a standard PRF, there is a secret function key fk that allows the holder to evaluate the function on any input x . However, there is also a public evaluation key ek , that allows one to evaluate the function on any input x belonging to some NP language L , provided the evaluator also has the corresponding witness w . The basic security notion says that for any $x \notin L$, the output of the function looks uniform even given the public evaluation key. A stronger *interactive* security variant says that the above should hold even if the adversary can query the function on arbitrary other inputs $x \notin L$. It is trivial to construct witness encryption from witness PRFs (with basic security), but the other direction is not known.

We show that UWM and one-way functions imply witness PRFs. In fact, we show that witness PRFs are equivalent to a weaker form of designated-verifier UWMs (dv -UWMs), where the public CRS is generated together with a secret verification key needed to verify proofs. In this case we can define two flavors of soundness: A basic soundness guarantee when the adversarial prover does not get any information about the secret verification key, beyond seeing the CRS, and a *reusable soundness* guarantee for dv -UWMs, where the adversarial prover can make verification queries to check whether purported proofs for various (true or false) statements x would be accepted using the verification key. We show that all four notions are equivalent in terms of feasibility (assuming one-way functions): witness PRFs with interactive security imply reusable dv -UWMs, which imply basic dv -UWMs, which imply basic witness PRFs, which then imply interactive witness PRFs. In particular, the last result shows that it is possible to generically upgrade witness PRFs with basic security to interactive security.

The above results place UWMs on the map somewhere between witness PRFs (which are equivalent to dv -UWMs) and iO . We also believe that UWMs are likely stronger than witness PRFs and dv -UWMs, mainly since we do not know of any way to generically go from the designated verifier setting to public verifiability. Moreover, we show that UWMs imply non-interactive zero-knowledge (NIZKs) with a deterministic prover, which are currently only known from iO , but not from witness PRFs.

Extreme Compactness implies Pseudo-Uniqueness. Next, we consider CWMs with “extreme compactness” $\alpha = O(\log \kappa)$ for security parameter κ . In other words, while we do not require the proofs to be unique, we require that the witness map can produce at most $2^\alpha = \text{poly}(\kappa)$ many possible proofs for each statement $x \in L$. We show that such extreme compactness is almost as good as uniqueness. In particular, we show that one can generically transform an extremely compact CWM into a pseudo-unique witness map (pseudo-UWM), where the pseudo-uniqueness property says the following. For any statement $x \in L$ and any two witnesses w_1, w_2 for x , both witnesses will map to the same “pseudo-unique” proof w^* with high $1 - 1/p(\kappa)$ probability over the choice of the CRS, where we can choose $p(\kappa)$ to be an arbitrarily large polynomial.

We note that a pseudo-UWM remains a powerful primitive. It can be used instead of UWM in applications where the “error” (i.e., non-uniqueness) can be “corrected.” In particular, it implies witness encryption. Indeed, in the construction of witness encryption using a UWM (from [4], or alternately, from a WPRF which is in turn constructed from a UWM as shown here), if we simply replace the UWM with a pseudo-UWM, it results in a small decryption error probability. This error probability can be made exponentially low by repeating the encryption process multiple times using independent keys and randomness, and during decryption, outputting the majority.

To show that extremely compact WMs imply pseudo-UWM, we solve an abstract problem of potentially independent interest that we refer to as *pseudo-deterministic sampling*. Consider a sampler that has oracle access to some arbitrary distribution \mathcal{D} whose support has polynomial size. The sampler can call the oracle polynomially many times and each call outputs a fresh random sample $x \leftarrow \mathcal{D}$. At the end, the sampler has to output some value x^* in the support of the distribution \mathcal{D} . Moreover, we want the sample x^* to be unique; if we run the sampler twice, with the oracle producing random/independent samples from \mathcal{D} in each run, the sampler should output the same value x^* in both executions with high $1 - 1/p(\kappa)$ probability. This guarantee is similar to pseudo-deterministic

algorithms [6, 8], which are randomized algorithms that nevertheless output a unique value independent of their randomness with high probability. We show how to solve the pseudo-deterministic sampling problem in the CRS model. The sample x^* that the sampler outputs may depend on the CRS but, with high probability, should be the same for every execution of the sampler with the given CRS, no matter what samples it receives from its oracle.

Mild Compactness implies NIZKs. We then turn our attention to CWMs with “mild compactness” where $\alpha = p(\kappa)$ for some fixed polynomial p , independent of the statement size $|x|$ or the witness size $|w|$. Such CWMs are implied by succinct non-interactive arguments (SNARGs) for NP, which are computationally sound proofs where the proof size is bounded by some fixed polynomial $p(\kappa)$, and independent of $|x|$ or $|w|$. The mild compactness of CWMs can be seen as a relaxation of the succinctness requirement for SNARGs, where the latter requires the proof to have small size $p(\kappa)$, while the former only requires the number of possible proofs that the prover outputs to be bounded by $2^{p(\kappa)}$ but allows the size of the proofs to be arbitrarily large. Although mildly compact CWMs are weaker than SNARGs, we show that they nevertheless imply non-interactive zero knowledge (NIZK) proofs. We generalize the recent work of [10], who showed how to construct NIZKs from SNARGs, by showing that the same result holds if we replace SNARGs by mildly compact CWMs. The above shows that mildly compact CWMs lie somewhere between NIZKs and SNARGs.

UWMs with Statistical Soundness and UP. Lastly, we ask whether we can get UWMs with statistical/perfect soundness. This appears highly unlikely since it would imply a construction of witness PRFs (and hence witness encryption) from one-way functions. But can we rule out this possibility under some well-studied complexity assumption? Interestingly, we do not know the answer to this question. Intuitively, we’d like to say that perfectly sound UWMs for NP would imply $\text{NP} = \text{UP}$, where UP is the class of languages where every statement x has a unique witness w^* . However, a perfectly sound UWM only guarantees that the prover outputs a unique proof and that the verifier never accepts a proof for a false statement, but it may still be possible for the verifier to accept many possible proofs besides the one that the prover outputs. We define the stronger notion of verifier-unique witness maps (VUWM) where we also guarantee that the verifier only accepts a unique proof w^* for each x , and show that perfectly sound VUWMs for NP imply $\text{NP} = \text{UP}$.

1.2 Technical overview

1.2.1 dv-UWM is equivalent to witness PRFs. To show the equivalence between dv-UWM and witness PRF wPRF, we first show that wPRF implies dv-UWM.

Witness PRF implies dv-UWM: This direction is rather straightforward and follows from the definition of wPRF. In particular, the dv-UWM proof w^* is computed by running the public evaluation algorithm using the evaluation key ek . The verification algorithm of dv-UWM is obtained by running the secret evaluation algorithm of wPRF using the secret function key fk and checking if the proof w^* is equal to the output of this algorithm. The correctness of the construction follows from the fact that the values computed in both the modes of wPRF are equal. Uniqueness is guaranteed since the private evaluation algorithm does not depend on the witness w and deterministically maps x to a unique output value. Finally, the soundness of dv-UWM follows from the interactive security of wPRF.

dv-UWM implies witness PRF: We show this result in two steps – (i) First, we show that a construction of non-interactive witness PRF for **NP** from any non-reusably sound dv-UWM for **NP**, (ii) next, we show a generic transformation from any non-interactive witness PRF for **NP** to an interactive witness PRF for **NP** additionally using one way function.

Non-reusably sound UWM implies non-interactive witness PRF: To construct a non-interactive wPRF from a (non-reusable) dv-UWM, the key generation algorithm wPRF.Gen of wPRF samples a random seed z for a (length-doubling) pseudorandom generator G and sets $y = G(z)$. It then runs the setup algorithm of DV-UWM, i.e., **dv.setup** to obtain $((K, VK))$. It then sets the evaluation key as $ek = (K, VK, y)$ and the function key as $fk = z$. To compute the function $F(fk, \cdot)$ on input $x \in L$, the evaluator uses DV-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 then outputs a hardcore bit (e.g., the Goldreich-Levin (GL) hardcore bit) of w^* as the pseudorandom bit b . In the public evaluation mode, on input (x, w) the algorithm wPRF.Eval uses the UWM to map the witness w for x into the unique witness w^* for the statement \hat{x} . It can then compute the pseudorandom bit b using the GL predicate. Intuitively, if an adversary can break wPRF security, then it can distinguish the bit 0 and 1 with non-negligible probability even if x is a false statement. This means that, using GL 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 dv-UWM.

Generic transformation from non-interactive to interactive witness PRF: To construct interactive witness PRF from non-interactive witness PRF (nl-wPRF) we need to carefully define the relation for the underlying nl-wPRF. In particular, the key generation algorithm of the interactive witness PRF **wPRF** runs the key generation algorithm of the non-interactive witness PRF **nl-wPRF** to obtain (\hat{ek}, \hat{fk}) . It then uses a statistically binding commitment scheme to commit to a message $\mathbf{0}$ (using randomness r) such that $\mathbf{0}$ is not a valid statement of the underlying **NP** relation \mathcal{R} for **wPRF** to obtain commitment c . In other words, $\mathbf{0} \notin \mathcal{X}$, where \mathcal{X} is the statement space of \mathcal{R} . It then sets the evaluation key as $ek = (\hat{ek}, c)$ and the function key as $fk = (\hat{fk}, r)$. To compute the function $F(fk, \cdot)$ on input $x \in L$, the evaluator uses **nl-wPRF** to get a value y for the statement \hat{x} stating that “either x is true or c is a commitment to some message x' such that $x \neq x'$ ”. In the public evaluation mode, on input (x, w) the algorithm **wPRF.Eval** uses the public evaluation key of the underlying **nl-wPRF** to map the witness w for x into the value y for the statement \hat{x} . In the proof, when the adversary commits to a challenge $x^* \notin L$, we switch from a commitment to $\mathbf{0}$ to a commitment c^* to x^* . Hence, we have that the statement $\hat{x} = (x^*, c^*)$ is now false. The hiding property of the commitment allows us to make such a switch. On the other hand, for all other statement $x_i \neq x^*$, the statement (x_i, c^*) is still true, and hence we can simulate the queries of the **wPRF** adversary using the function $\text{nl-wPRF.F}(\hat{fk}, \cdot)$.

1.2.2 Extremely Compact WM implies Pseudo-Unique WM.

As mentioned above, to construct a Pseudo-UWM from an extremely compact WM, we solve the abstract problem of *pseudo-deterministic sampling* from a distribution with polynomial-sized support. We briefly sketch our solution to the pseudo-deterministic sampling problem.

Solving Pseudo-Deterministic Sampling. As a first attempt, consider obtaining N samples from the distribution for a value N that is much larger than the size of the distribution’s support, and then taking the lexicographically smallest one. This would indeed work if we could ensure that every element in the support gets sampled at least once. Unfortunately, this does not hold true for arbitrary distributions. For instance, if the lexicographically smallest element in the support has a probability $1/N$, then there is a constant probability for it to get sampled as well as to not get sampled. As a second attempt, one may consider using a hash

function to define the lexicographic ordering to prevent the distribution from adversarially assigning such a probability to the smallest element; however, this does not help either, if a large fraction of the elements in the support have probability $1/N$. One may note that the difficulty here arises from (moderately) low probability elements; so to avoid such elements, we could try to pick a high probability element, which is guaranteed to occur many times in the sample. However neither picking the most frequent element (e.g., when there are multiple elements which have the maximum probability), nor picking the lexicographically smallest one from among a selected subset of frequent elements (e.g., when there are elements with probabilities that place them near the threshold used for selection) is sufficient to guarantee uniqueness. Our final solution combines ideas from all of these approaches: it obtains N samples and would choose one with the smallest hash value, but the hash is computed on *the element concatenated with a counter*. That is, if an element x occurs k times in the sample, then the hashes of all of $x||1, x||2, \dots, x||k$ are considered. This has the effect of picking an element from among the more frequent elements, but without creating a threshold for being considered frequent. Using elementary concentration bounds we show that the probability of two executions of this process yielding different outcomes (when using a N -wise independent hash function) goes down polynomially with $1/N$.

Pseudo-UWM from Pseudo-Deterministic Sampling. To reduce pseudo-UWM to pseudo-deterministic sampling, first we need to create a distribution over proofs that remains (essentially) the same for all witnesses. We achieve this using a Non-Interactive Witness Indistinguishable proof system (NIWI). Then we map this NIWI proof using the extremely compact WM (for the relation corresponding to NIWI verification) into a polynomial-sized support. The soundness of this proof depends on the fact that for any false statement, there should not *exist* a NIWI proof that gets accepted; this is guaranteed by using a statistically sound NIWI. At this point, we have a proof system that is sound, compact, and witness-indistinguishable. Now, pseudo-deterministic sampling from this distribution would result in a pseudo-UWM.

1.2.3 Mildly CWM implies NIZK. We show that CWM with compactness level $\alpha = \text{poly}(\kappa)$ for some fixed polynomial $\text{poly}(\cdot)$, independent of the statement size $|x|$ or the witness size $|w|$ implies the existence of NIZK argument system. Our construction generalizes the recent work of [10] who constructed NIZKs from SNARG by replacing SNARG with

CWM with the above compactness level. [10] shows how to compile any NIZK in the hidden-bits model (HBM) to NIZK in the CRS model using a primitive called *hidden-bits generator with subset-dependent proofs* (SDP-HBG). Then they show how to construct such a SDP-HBG from any SNARG and bounded-leakage weak PRF (BLR-wPRF)⁴. Below we sketch the main idea of the construction and the proof of [10]. We then show how to modify their construction and proof technique when using CWM instead of SNARG. A SDP-HBG consists of the following algorithms:

- $\text{HBG}^{\text{sdp}}.\text{Setup}(1^\kappa, 1^n)$ generates a CRS crs where n denotes the length of hidden-bits to be generated.
- $\text{HBG}^{\text{sdp}}.\text{GenBits}(\text{crs})$ generates “hidden-bits” $r \in \{0, 1\}^n$ and a state st .
- $\text{HBG}^{\text{sdp}}.\text{Prove}(\text{st}, I)$ generates a proof π that certifies the sub-string r_I .
- $\text{HBG}^{\text{sdp}}.\text{Verify}(\text{crs}, I, r_I, \pi)$ verifies the proof π to ensure that the sub-string of r on the positions corresponding to subset I is indeed r_I .

The SDP-HBG is required to satisfy the following properties – (i) *Somewhat Computational Binding*, which requires that exists a “sparse” subset $\mathcal{V}^{\text{crs}} \in \{0, 1\}^n$ of size much smaller than 2^n such that no PPT malicious prover can generate a proof for bits that are not consistent with any element of \mathcal{V}^{crs} , (ii) *Hiding*, which requires that for any subset $I \subseteq [n]$, no PPT adversary given can distinguish $r_{\bar{I}}$ from a uniformly random string $r'_{\bar{I}}$, where $r_{\bar{I}}$ denotes the substring of r on the positions corresponding to $\bar{I} = [n] \setminus I$. To construct SDP-HBG, the setup algorithm $\text{HBG}^{\text{sdp}}.\text{Setup}(1^\kappa, 1^n)$ samples $\vec{x} = (x_1, \dots, x_n) \in \{0, 1\}^{m \times n}$ and sets $\text{crs} = \vec{x}$. The algorithm $\text{HBG}^{\text{sdp}}.\text{GenBits}(\text{crs})$ derives the hidden bits $\vec{r} = (r_1, \dots, r_n) \in \{0, 1\}^n$ as $r_i = F_K(x_i)$, where $F_K : \{0, 1\}^m \rightarrow \{0, 1\}$ is a λ -BLR-wPRF and $K \in \{0, 1\}^k$ for some polynomial $k = k(\kappa, \lambda)$, where κ is the security parameter. The algorithm $\text{HBG}^{\text{sdp}}.\text{Prove}(\text{st}, I)$ then uses the SNARG to generate a proof π for the statement that the values r_i for all $i \in [I]$ are correctly computed, using K as the witness. The verification then consists of verifying the SNARG proof.

The somewhat computational binding property of SDP-HBG easily follows from the soundness of SNARG as long as $k \ll n$ (where k is the size of the PRF key K and n is the length of the hidden bit string). The hiding property is easy to reduce to security of the underlying BLR-wPRF

⁴ Informally, a λ -BLR-wPRF $F_K(\cdot)$ guarantees pseudorandomness of the output of the PRF when evaluated on uniformly random inputs, even when the adversary can leak up to λ bits on K . [9] showed how to construct such BLR-wPRF from any OWF.

as long as $|\pi| \leq \lambda$. In particular, the proof π corresponds to the subset I which does not depend on $x_{\bar{I}}$ (the bits of x in the positions $[n] \setminus I$), and thus we can think of $x_{\bar{I}}$ as the challenge inputs and π as the leakage.

Using CWM instead of SNARG. Our construction follows the same blueprint from [10], except that we use a α -CWM for $\alpha = \text{poly}(\kappa)$ instead of a SNARG to generate the proof π and use an entropic leakage-resilient weak PRF⁵ to generate the hidden bits $r \in \{0, 1\}^n$, instead of a bounded leakage-resilient weak PRF. This is because, unlike SNARG the size of our CWM proofs π are *not* guaranteed to be succinct. However, we have the guarantee that the proof π is α -compact, for some $\alpha = \text{poly}(\kappa)$, where $\text{poly}(\kappa)$ is independent of n . This means the size of the CWM image is at most 2^α . Hence, as long as the underlying wPRF is resilient to λ -entropic leakage, where $\alpha \leq \lambda$, we can rely on the (entropic) leakage-resilience of ELR-wPRF to argue hiding of the SDP-HBG. We can then set the parameters appropriately to satisfy these two inequalities.

1.2.4 UWM implies Deterministic-prover NIZK. We show that UWM implies deterministic prover NIZK arguments systems (DP-NIZK), where the prover and verifier are deterministic. In fact, we can achieve *perfect* zero-knowledge property. The main idea of our construction is similar to the construction of non-interactive witness PRF from dv-UWM. In particular, the setup algorithm of DP-NIZK chooses a pseudorandom string $y = G(z)$, where G is a length-doubling PRG. The CRS crs of DP-NIZK consists of the CRS K of UWM and the value y . The prover of DP-NIZK on input some (x, w) in the relation runs the UWM prover to get a representative witness w^* for the statement \hat{x} stating that “either x is true or y is pseudorandom”, using w as the witness. Note that the prover is deterministic. The verification of DP-NIZK simply uses the UWM verifier. To prove the soundness of this construction, we sample y uniformly at random and hence with very high probability there does not exist any valid preimage of y with respect to G . Hence, if $x \notin L$, the statement $\hat{x} = (x, y)$ is also not in the (augmented) language with overwhelming probability. The soundness of the construction now follows from the soundness of UWM. To prove zero-knowledge property, the simulator uses the trapdoor z as the witness to simulate proofs of statements $x_i \in L$ queried by the adversary (note that z is a valid witness

⁵ Informally, a λ -entropic leakage-resilient PRF $F_K(\cdot)$ guarantees pseudorandomness of the output of the PRF when evaluated on uniformly random inputs, even when the adversary can get λ -entropic leakage on K . Roughly this means that the PRF key K still has $k - \lambda$ bits of average min-entropy, even conditioned on the leakage from K . [9] showed how to construct such entropic LR-wPRF from any OWF.

for the statements (x_i, y)). The uniqueness property of UWM guarantees that proofs computed by either of the witnesses result in the same proof. Hence, the zero-knowledge property follows. Finally, note that, we can achieve the stronger notion of perfect ZK since the CRS in both the real world and the simulation are identically distributed (both are computed by sampling the CRS K of UWM and the string y pseudorandomly).

1.2.5 Perfectly Sound Verifier UWM implies $\mathbf{NP} = \mathbf{UP}$. Recall that a verifier UWM (VUWM) is similar to an UWM with the additional guarantee that the verifier also accepts a unique proof for each statement x . The complexity class \mathbf{UP} consists of problems that are accepted by an unambiguous Turing machine with at most one accepting path for each input. It is easy to see that the verifier of a perfectly sound VUWM acts as a \mathbf{UP} relation. Also, since we require the VUWM to be perfectly sound it does not require as setup. Hence, it shows that $\mathbf{NP} \subseteq \mathbf{UP}$. The other direction is trivial and hence this shows that $\mathbf{NP} = \mathbf{UP}$.

2 Preliminaries

2.1 Notation

For $n \in \mathbb{N}$, we write $[n] = \{1, 2, \dots, n\}$. 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 $\text{poly}(|x|)$ 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 . We refer the reader to the full version of the paper for some basic definitions and concepts related to information theory. We will need the following form of the Chernoff-Hoeffding inequality.

Lemma 1. *Let S_N be the sum of N independent samples of a Bernoulli random variable, which is 1 with probability p and 0 otherwise. Then $\Pr[|S_N - Np| > t] < e^{-t^2/N}$. In particular, $\Pr[|S_N - Np| > N^{2/3}] < e^{-N^{1/3}}$.*

2.2 Cryptographic Primitives

Next we summarize some of the cryptographic primitives from literature that we rely on.

2.2.1 Witness PRFs A witness PRF [17] consists of triple of algorithms $\text{wPRF} = (\text{wPRF.Gen}, F, \text{wPRF.Eval})$ as follows:

1. $\text{wPRF.Gen}(\kappa, R)$: This is a randomized algorithm that takes as input the security parameter 1^κ and the description of a circuit $R : \mathcal{X} \times \mathcal{W} \rightarrow \{0, 1\}$ and outputs a function secret key fk along with a public evaluation key ek .
 2. $F(\text{fk}, x)$: The private evaluation algorithm F is a deterministic algorithm that takes as input the function secret key fk and an input $x \in \mathcal{X}$ and produces some output $y \in \mathcal{Y}$ for some set \mathcal{Y} .
 3. $\text{wPRF.Eval}(\text{ek}, x, w)$: The public evaluation algorithm wPRF.Eval is also a deterministic algorithm that takes as input the public evaluation key ek , an input $x \in \mathcal{X}$ and a witness $w \in \mathcal{W}$ to produce an output $y \in \mathcal{Y}$ or \perp .
- * *Correctness.* The correctness of wPRF requires that for all $x \in \mathcal{X}$ and $w \in \mathcal{W}$, the following holds:

$$\text{wPRF.Eval}(\text{ek}, x, w) = \begin{cases} F(\text{fk}, x) & \text{if } R(x, w) = 1 \\ \perp & \text{if } R(x, w) = 0 \end{cases}$$

* *Security.* We recall the *adaptive instance interactive security* notion for witness PRFs from [17]. Consider the following experiment $\text{Exp}_{\mathcal{A}}^R(\kappa, b)$ between an adversary \mathcal{A} and a challenger \mathcal{C} , parameterized by a relation $R : \mathcal{X} \times \mathcal{W} \rightarrow \{0, 1\}$, a bit b and security parameter κ .

- The challenger \mathcal{C} runs $(\text{fk}, \text{ek}) \leftarrow \text{wPRF.Gen}(\kappa, R)$, and gives ek to \mathcal{A} .
- \mathcal{A} can adaptively make queries on instances $x_i \in \mathcal{X}$ and receives the values $F(\text{fk}, x_i)$ from \mathcal{C} .
- At any point in the game, \mathcal{A} can make a challenge query $x^* \in \mathcal{X}$. The challenger computes $y_0 \leftarrow F(\text{fk}, x^*)$ and $y_1 \xleftarrow{\$} \mathcal{Y}$. It then returns y_b to \mathcal{A} .
- \mathcal{A} can make additional queries to F and finally \mathcal{A} outputs a bit b' . The challenger \mathcal{C} checks that $x^* \notin \{x_i\}$ and that $x^* \notin L_R$ ⁶. If either check fails, \mathcal{C} outputs a random bit. Otherwise, it outputs b' .

Let W_b be the event that the challenger in experiment $\text{Exp}_{\mathcal{A}}^R(\kappa, b)$ outputs 1. Define the advantage of \mathcal{A} as $\text{wPRF.Adv}_{\mathcal{A}}^R(\kappa) = |\Pr[W_0] - \Pr[W_1]|$.

⁶ Note that, the language $L_R := \{x \mid \exists w, (x, w) \in R\}$

Definition 1 (Adaptive instance Interactive security). $\text{wPRF} = (\text{wPRF.Gen}, F, \text{wPRF.Eval})$ is adaptive instance interactively secure for a NP relation R if for all PPT adversaries \mathcal{A} , the advantage $\text{wPRF.Adv}_{\mathcal{A}}^R$ of \mathcal{A} is negligible in the security parameter κ .

One can also define *non-interactive* security for witness PRFs, where the adversary \mathcal{A} in the above experiment is not allowed to make any F queries.

Definition 2 (Adaptive instance Non-Interactive security). $\text{wPRF} = (\text{wPRF.Gen}, F, \text{wPRF.Eval})$ is adaptive instance non-interactively secure for a NP relation R if for all PPT adversaries \mathcal{A} , the advantage $\text{wPRF.Adv}_{\mathcal{A}}^R$ of \mathcal{A} is negligible in the security parameter κ , and additionally the adversary \mathcal{A} is not allowed to make any F queries in the above experiment.

Finally, one can also define a weaker notion of security, called the *static instance interactive (non-interactive) security*, where the adversary needs to commit to the challenge x^* before seeing ek or before making any queries to the oracle F . We note that one can convert any static-instance interactive (resp. non-interactive) witness PRF to an adaptive instance one by relying on complexity leveraging when appropriate.

2.2.2 Generalized Goldreich-Levin Theorem. For our construction of witness PRF from witness maps we will need to use a generalized version of the Goldreich-Levin (GL) theorem [7], as stated below.

Lemma 2 (Generalized Goldreich-Levin Theorem). *There exists a PPT inverter \mathcal{A}' and a non-zero polynomial $q(\cdot)$ such that, for any PPT algorithm \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 $\Pr[\mathcal{A}'^{\mathcal{A}(\alpha, \cdot)}(1^\ell, \alpha) = \beta] \geq q(p(\alpha) - \frac{1}{2})$.*

2.2.3 Leakage-resilient weak PRF. A standard weak PRF (wPRF) requires that given arbitrarily many uniformly random inputs x_1, \dots, x_q , the outputs of the wPRF y_1, \dots, y_q look pseudorandom. A leakage-resilient wPRF (LR-wPRF) requires wPRF security to hold even if the attacker can leak some information about the secret key. In particular, we will consider the entropy-bounded leakage model [3, 5]. Following [5], we first recall the notion of λ -leaky functions.

Definition 3 (λ -leaky function). A probabilistic function $h : \{0, 1\}^* \rightarrow \{0, 1\}^*$ is λ -leaky if, for all $n \in \mathbb{N}$ we have $\tilde{H}_\infty(\mathcal{U}_n | h(\mathcal{U}_n)) \geq n - \lambda$, where \mathcal{U}_n is the uniform distribution over $\{0, 1\}^n$.

As shown in [5], if a function is λ -leaky (decreases the entropy of the uniform distribution by at most λ bits), then it decreases the entropy of every distribution by at most λ bits. Moreover, the definition composes nicely and an adversary that adaptively chooses several λ_i -leaky functions, only learns $\sum_i \lambda_i$ bits of information.

Informally, we say a function $\mathcal{F}_K(\$)$ is a $\lambda(\kappa)$ -leakage-resilient weak PRF in the entropy-bounded leakage model, if the weak PRF security guarantee is maintained, even if the adversary can learn the output of a λ -leaky function on the key K . We refer to the full version for the formal definition.

We also rely on statistically-sound NIWI proof systems. We refer to the full version for the formal definition.

3 Different notions of Witness Maps and their definitions.

In this section, we present the definition of compact witness map from [4]. We then present different variations of their definition, namely designated-verifier witness maps and verifier-compact witness maps, which we introduce in this work. We start by recalling the definition of compact witness map (CWM) from [4].

3.1 Compact Witness Maps.

We say 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 4 (Compact Witness Map (CWM)). *For $\alpha \geq 0$, an α -CWM for an **NP** relation R is a triple $\text{CWM} = (\text{setup}, \text{map}, \text{check})$ where setup is a PPT algorithm and the other two are deterministic polynomial time algorithms such that:*

- $\text{setup}(\kappa, \ell)$ outputs a string K of length polynomial in the security parameter κ and ℓ , where $\ell = \ell(\kappa)$ is an upper bound on the length of the statements supported by CWM.
- Completeness: For any polynomial ℓ , $\forall (x, w) \in R_{\ell(\kappa)}, \forall K \leftarrow \text{setup}(\kappa, \ell(\kappa))$,

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

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

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

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

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

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

The above definition has perfect security in the sense that the completeness and compactness 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.

3.2 Designated-Verifier Witness Maps.

In this section, we define (reusable/non-reusable) designated-verifier compact witness maps (dv-CWM).

Definition 5 (Reusable/Non-Reusable dv-CWM). For $\alpha \geq 0$, a reusable (resp. non-reusable) α -dv-CWM for an NP relation R is a triple $\text{DV-CWM} = (\text{setup}, \text{map}, \text{check})$ where setup is a PPT algorithm that on input the security parameter κ and the statement length ℓ , outputs a pair of strings (K, VK) , and the other two are deterministic polynomial time algorithms that satisfy the completeness, compactness and reusable soundness (resp. non-reusable soundness) conditions below.

- Completeness: For any polynomial ℓ , $\forall (x, w) \in R_{\ell(\kappa)}$, $\forall (K, VK) \leftarrow \text{setup}(\kappa, \ell(\kappa))$,

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

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

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

- Reusable/Non-Reusable Soundness: Reusable soundness requires that the advantage $\text{Adv}_{\mathcal{A}}^{\text{DV-CWM}}(\kappa)$ defined below is negligible for every polynomial ℓ and every PPT adversary \mathcal{A} with oracle access to $\text{check}(VK, \cdot, \cdot)$.

$$\text{Adv}_{\mathcal{A}}^{\text{DV-CWM}}(\kappa) = \Pr_{\substack{(K, VK) \leftarrow \text{setup}(\kappa, \ell(\kappa)) \\ (x^*, w^*) \leftarrow \mathcal{A}^{\text{check}(VK, \cdot, \cdot)}(K)}} [\text{check}(VK, x^*, w^*) = 1 \wedge x^* \notin L_R].$$

Non-reusable soundness requires that $\text{Adv}_{\mathcal{A}}^{\text{DV-CWM}}(\kappa)$ is negligible for every polynomial ℓ and every PPT adversary \mathcal{A} which does not access its oracle.

A 0-dv-CWM is also called a Designated-verifier UWM (dv-UWM).

Similar to CWM, one can also define a weaker notion of *selective reusable (resp. non-reusable) soundness*, in which the adversary is required to generate x^* first (given κ, ℓ) before it gets K and access to the verification oracle $\text{check}(VK, \cdot, \cdot)$.

3.3 Verifier-Compact Witness Maps.

Definition 6 (Verifier-Compact Witness Maps (VCWM)). *For $\alpha \geq 0$, an α -VCWM ($\text{setup}, \text{map}, \text{check}$) for an **NP** relation R is a CWM for R satisfying the following additional condition:*

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

$$|\{w^* \mid \text{check}(K, x, w^*) = 1\}| \leq 2^\alpha.$$

A 0-VCWM is also called a verifier-unique witness map (VUWM).

Selective Soundness. The soundness condition for CWM and its variants (dv-CWM and VCWM) can be relaxed to obtain a *selectively sound* variant of the corresponding primitive. In the soundness conditions above, we considered an adversary \mathcal{A} which outputs a statement x^* and a purported proof w^* at the end of the experiment. For selective soundness, we require \mathcal{A} to output x^* at the beginning (given only κ, ℓ), before setup is executed. This level of soundness suffices for some applications (e.g., construction of a witness encryption scheme from a UWM), as shown in [4]. 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.

4 Equivalence of Witness PRFs and dv-UWM

In this section, we explore the relationship between witness PRF (wPRF) and unique witness maps. In particular, we show that witness PRF and designated-verifier UWM (dv-UWM) are equivalent for **NP**. For these implications, we consider the static security variant of witness PRF and selective soundness of dv-UWM. Our implications can be adapted to the adaptive security variants of both these notions via complexity leveraging.

4.1 Witness PRF imply dv-UWM

In this section, we present the construction of our (selective reusable sound) dv-UWM for any **NP** relation R . The main building block of our construction is a (static-instance secure) witness PRF for R .

Construction. Let $\text{wPRF} = (\text{wPRF.Gen}, F, \text{wPRF.Eval})$ be a *static-instance interactively secure* witness PRF for any **NP** relation R parametrized by statements of length at most $\ell(\kappa)$, where κ is the security parameter and ℓ is an arbitrary (but fixed) polynomial in the security parameter. We construct a (selective) *reusable* dv-UWM $\text{DV-UWM} = (\text{dv.setup}, \text{dv.map}, \text{dv.check})$ for R as follows:

- $\text{dv.setup}(\kappa, \ell) : \text{Run } (\text{fk}, \text{ek}) \leftarrow \text{wPRF.Gen}(\kappa, R)$. Set $K = \text{ek}$ and $VK = \text{fk}$.
- $\text{dv.map}(K, x, w) : \text{Parse } K \text{ as } \text{ek}$. Run $y = \text{wPRF.Eval}(\text{ek}, x, w)$. Output $w^* = y$
- $\text{dv.check}(VK, x, w^*) : \text{Parse } VK \text{ as } \text{fk}$ and compute $y' = F(\text{fk}, x)$ and check if $w^* \stackrel{?}{=} y'$. If the check is satisfied, output 1; else output \perp .

Theorem 1. *Let wPRF be an static-instance interactively secure witness PRF for **NP** with super-polynomial range $|\mathcal{Y}| = \kappa^{\omega(1)}$. Then the above construction of dv-UWM for **NP** satisfies selective reusable soundness.*

Proof. Firstly, we note that DV-UWM satisfies perfect completeness (assuming wPRF is perfectly correct). Also, it satisfies uniqueness, since (x, w) is deterministically mapped to the output of wPRF , regardless of the witness w . In particular, the correctness of wPRF guarantees that for all $(x, w) \in R$, $\text{wPRF.Eval}(\text{ek}, x, w) = F(\text{fk}, x)$. The later function (i.e., $F(\text{fk}, \cdot)$) does not depend on w and deterministically maps x to a unique output value $y \in \mathcal{Y}$. Below, we shall prove that the construction satisfies selective reusable soundness as well.

Consider an adversary \mathcal{A} in the definition of $\text{Adv}_{\mathcal{A}}^{\text{DV-UWM}}(\kappa)$ (see Def. 5). Note that, in the (selective) reusable soundness experiment the adversary \mathcal{A} first commits to the challenge x^* and then gets access to the public parameter K and the verification oracle, namely $\text{dv.check}(VK, \cdot, \cdot)$. The oracle takes as input tuples of the form (x_i, w_i^*) and outputs either 1 or 0. We show how to construct another adversary \mathcal{B} breaking the static-instance interactive security of wPRF using \mathcal{A} in a black-box way. The adversary \mathcal{B} simulates the environment of \mathcal{A} as follows:

1. The adversary \mathcal{B} first commits to a challenge x^* such that $x^* \notin L_R$. The adversary \mathcal{A} forwards x^* to its own challenger and receives a value $y^* \in \mathcal{Y}$, which it stores in its memory.

2. The adversary \mathcal{B} then receives ek from its challenger and sets $K = \text{ek}$. It gives K to \mathcal{A} .
3. When \mathcal{A} queries a tuple (x_i, w_i^*) to $\text{dv.check}(K, \text{VK}, \cdot, \cdot)$, the adversary \mathcal{B} does the following:
 - Query the oracle $F(\text{fk}, \cdot)$ on input x_i to receive some output y_i .
 - Checks if $w_i^* \stackrel{?}{=} y_i$. If so, it outputs 1 to \mathcal{A} else it outputs 0.
4. Finally, at some point \mathcal{A} outputs w^* corresponding to the challenge x^* (which it committed to before). The adversary \mathcal{A} then retrieves the value y^* from its memory and checks if $y^* \stackrel{?}{=} w^*$. If the check passes, \mathcal{B} outputs 0; else it outputs 1.

This completes the description of simulation of \mathcal{A} 's environment by \mathcal{B} . Let us assume that \mathcal{A} makes a total of $q = q(\kappa)$ queries to the verification oracle $\text{dv.check}(\text{VK}, \cdot, \cdot)$ (including the challenge query). Since \mathcal{A} has some non-negligible advantage (say ϵ) in breaking the selective reusable soundness of DV-UWM, it must hold that $\text{dv.check}(\text{VK}, x^*, w^*) = 1$ holds with probability ϵ . According to the construction, the above check passes whenever $w^* = F(\text{fk}, x^*)$. If the value y^* received by \mathcal{B} from its challenger was computed using the function $F(\text{fk}, \cdot)$, then it always holds that $y^* = w^*$. However, if y^* was randomly sampled, the probability that w^* is equal to y^* is $\frac{1}{|\mathcal{Y}|}$, which is negligible. Hence, the advantage of \mathcal{B} in breaking the adaptive-instance interactive security of wPRF is $\epsilon - \frac{q}{|\mathcal{Y}|}$, which is non-negligible, thereby contradicting the security of wPRF. \square

4.2 dv-UWM implies Witness PRF

Now, we present our implication in the other direction, namely that dv-UWM for **NP** implies witness PRF for **NP**. We split our transformation into two phases. First, we present a construction of a (static) non-interactive witness PRF for **NP** from any (selective) non-reusably sound dv-UWM for **NP**. Next, we show a generic transformation from any (static) non-interactive witness PRF for **NP** to an (static) interactive witness PRF for **NP** additionally using any one way function.

4.2.1 dv-UWM implies Non-interactive Witness PRF. In this section, we show our first transformation from any (selective) non-reusably sound dv-UWM for **NP** to a (static-instance) non-interactive witness PRF for **NP**. The construction is shown in [Figure 2](#).

Theorem 2. *If DV-UWM is a selective non-reusably sound dv-UWM for the **NP** relation \mathcal{R} (defined in [Figure 2](#)), then the construction shown in [Figure 2](#) is a static-instance non-interactively secure witness PRF for the **NP** relation \mathcal{R} .*

Let R be a **NP** relation, and L_R be the corresponding **NP** language defined as $L_R := \{x \mid \exists w : (x, w) \in R\}$. Let R' be another **NP** relation defined as $(y, z) \in R'$ if and only if $y = G(z)$, where $G : \{0, 1\}^\kappa \rightarrow \{0, 1\}^{2\kappa}$ is a length-doubling pseudo-random generator. Also, let $L_{R'}$ be the corresponding **NP** language defined as $L_{R'} := \{y \mid \exists z : (y, z) \in R'\}$. Further, we assume that R and R' are parameterized with their input lengths. Define the following derived **NP** relation \mathfrak{R} and language $L_{\mathfrak{R}}$ as:

$$\mathfrak{R}((x, y), (w, z)) = 1 \iff R(x, w) = 1 \vee R'(y, z) = 1, \text{ and}$$

$$L_{\mathfrak{R}} = \{(x, y) \mid \exists (w, z), ((x, y), (w, z)) \in \mathfrak{R}\}.$$

Note that, the relation \mathfrak{R} is parameterized with statements of length at most $\ell' = \ell + 2\kappa$.

- (a) Let $\text{DV-UWM} = (\text{dv.setup}, \text{dv.map}, \text{dv.check})$ be a (selectively) sound dv-UWM for the language $L_{\mathfrak{R}}$. Further, let the length of the representative w^* of DV-UWM be $p(\kappa)$ bits, for some polynomial $p(\cdot)$.
 - (b) Let $\text{GL}(\pi, r)$ denote the Goldreich-Levin (GL) hardcore bit [7] of π using randomness r . Recall that, the GL predicate is the bit-wise inner product of π and r .
1. $\text{wPRF.Gen}(\kappa, R)$: Takes as input an **NP** relation R (parametrized by its input length ℓ) as defined above. Run $(K, VK) \leftarrow \text{dv.setup}(\kappa, \ell')$, where ℓ' is defined as above. Sample $z \leftarrow \{0, 1\}^\kappa$ and $r \leftarrow \{0, 1\}^{p(\kappa)}$ uniformly at random, and compute $y = G(z)$. Set $\text{ek} = (K, y, r)$ and $\text{fk} = z$.
 2. $\text{F}(\text{fk}, x)$: Takes as input an instance $x \in L_R$. It does the following:
 - Computes the representative witness $w^* = \text{dv.map}(K, (x, y), (\perp, z))$, using (\perp, z) as witness. Note that, $(x, y) \in L_{\mathfrak{R}}$.
 - Compute the GL hardcore bit $b = \text{GL}(w^*, r)$.
 3. $\text{wPRF.Eval}(\text{ek}, x, w)$: Takes as input an instance $x \in L_R$. It does the following:
 - Computes a representative witness $w^* = \text{dv.map}(K, (x, y), (w, \perp))$, using (w, \perp) as witness. Note that, $(x, y) \in L_{\mathfrak{R}}$.
 - Compute the GL hardcore bit $b = \text{GL}(w^*, r)$.

Fig. 2. Construction of Non-Interactive Witness PRF **wPRF** from DV-UWM

Proof. We show that any adversary $\mathcal{A}_{\text{wprf}}$ breaking the static-instance non-interactive security of **wPRF** with a noticeable advantage can be transformed into an adversary $\mathcal{A}_{\text{dv-uwm}}$ breaking the selective non-reusable soundness of DV-UWM. Note that, in the static-instance non-interactive security game of **wPRF** the adversary commits to the challenge x^* before seeing the evaluation key ek . At first, we show that the adversary $\mathcal{A}_{\text{wprf}}$ breaking the security of **wPRF** can be converted into a predictor \mathcal{A}_{GL} for the generalized Goldreich-Levin theorem. In more details, let the adversary $\mathcal{A}_{\text{wprf}}$ can predict the bit b in the security experiment of **wPRF** on the challenge instance $x^* \notin L_R$ with non-negligible probability. Let, $\alpha = (K, y)$,

and $\beta = w^*$ from the generalized GL theorem (see [Lemma 2](#)). This implies that we can construct a distinguisher \mathcal{A} that on input $(\alpha = (K, y), r)$ can distinguish the bit b (in the above construction) from a random bit with non-negligible probability. Hence, by [Lemma 2](#), we can use this distinguisher \mathcal{A} to construct a predictor \mathcal{A}' , who given $\alpha = (K, y)$ can predict the pre-image w^* with non-negligible probability. This implies that the predictor outputs w^* such that $w^* = \text{dv.map}(K, (x, y), (\perp, z))$ with non-negligible probability. At this point, instead of computing $y = G(z)$, we sample a random $y \leftarrow \{0, 1\}^{2\kappa}$. 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 $\text{dv.check}(K, VK, (x, y), w^*) = 1$) continues to hold, except with a negligible probability. However, note that, with very high probability it holds that $(x, y) \notin \mathcal{R}$. This contradicts the selective non-reusable soundness property of dv-UWM, since the adversary \mathcal{A}' outputs a valid representative witness corresponding to a false statement $(x, y) \notin \mathcal{R}$. \square

4.2.2 Equivalence of Non-Interactive and Interactive Witness PRF. In this section, we show that any (static-instance) non-interactive witness PRF for **NP** can be generically transformed to a (static-instance) interactive witness PRF for **NP**. The construction is given in [Figure 3](#).

Theorem 3. *Let nl-wPRF be a static-instance non-interactively secure witness PRF for the **NP** relation Ψ (defined in [Figure 3](#)), and (Com, Open) be a statistically binding commitment scheme. Then the construction shown in [Figure 3](#) is a static-instance interactively secure witness PRF for the **NP** relation \mathcal{R} .*

The detailed proof of this theorem is presented in the full version of the paper. The intuition behind the proof is given in the technical overview.

5 Extremely Compact WM implies Pseudo-UWM

In this section, we show that an *extremely compact* WM – i.e., a CWM with polynomial-sized image – implies a Pseudo-UWM (p-UWM) where the uniqueness may not hold with an inverse polynomial probability (that can be made arbitrarily small). In other words, we show that a α -CWM for $\alpha = O(\log \kappa)$ for security parameter κ implies a p-UWM as defined below.

Definition 7 (Pseudo Unique Witness Map (p-UWM)). *A Pseudo-UWM (p-UWM) for an **NP** relation R is a triple $\text{CWM} = (\text{setup}, \text{map}, \text{check})$*

Let $\mathcal{R} : \mathcal{X} \times \mathcal{W} \rightarrow \{0, 1\}$ be an **NP** relation restricted to inputs in $\mathcal{X} = \{0, 1\}^{\ell(\kappa)}$, and $L_{\mathcal{R}}$ be the corresponding **NP** language defined as $L_{\mathcal{R}} := \{x \mid \exists w, (x, w) \in \mathcal{R}\}$. Let \mathcal{R}' be another **NP** relation defined as:

$$((x, c), (x', r)) \in \mathcal{R}' \iff (c = \text{Com}(x'; r) \wedge (x \neq x')),$$

where **Com** is a polynomial-time statistically binding commitment scheme over a message space $\mathcal{X} \cup \{\mathbf{0}\}$, where $\mathbf{0} \notin \mathcal{X}$. Also, let $L_{\mathcal{R}'}$ be the corresponding **NP** language. Finally, let us define the following **NP** relation Ψ and language L_{Ψ} as:

$$((x, c), (w, x', r)) \in \Psi \iff (x, w) \in \mathcal{R} \vee ((x, c), (x', r)) \in \mathcal{R}' \text{ and}$$

$$L_{\Psi} = \{(x, c) \mid \exists (w, x', r) \text{ s.t. } ((x, c), (w, x', r)) \in \Psi\}.$$

Note that, the relation Ψ is parameterized with the input length $\ell' = \ell + |c|$.

- Let **nl-wPRF** = (**nl-wPRF.Gen**, **nl-wPRF.F**, **nl-wPRF.Eval**) be a static-instance non-interactive witness PRF for the **NP** relation Ψ defined above.

We construct a static-instance interactively secure witness PRF **wPRF** = (**wPRF.Gen**, **F**, **wPRF.Eval**) as follows:

1. **wPRF.Gen**(κ, \mathcal{R}) : Takes as input the **NP** relation \mathcal{R} defined above. Run $(\hat{\text{ek}}, \hat{\text{fk}}) \leftarrow \text{nl-wPRF.Gen}(\kappa, \Psi)$, where the relation Ψ is defined as above. Sample a random tape r for the commitment scheme **Com**, and compute $c = \text{Com}(\mathbf{0}; r)$. Set $\text{ek} = (\hat{\text{ek}}, c)$ and $\text{fk} = (\hat{\text{fk}}, r)$.
2. **F**(fk, x) : Parse fk as $(\hat{\text{fk}}, r)$, compute the commitment $c = \text{Com}(\mathbf{0}; r)$ and output $y = \text{nl-wPRF.F}(\hat{\text{fk}}, (x, c))$.
3. **wPRF.Eval**(ek, x, w) : Takes as input an instance-witness pair $(x, w) \in \mathcal{R}$ and does the following: Parse ek as $(\hat{\text{ek}}, c)$ and output $\text{nl-wPRF.Eval}(\hat{\text{ek}}, (x, c), (w, \perp, \perp))$. Note that $(x, c) \in L_{\Psi}$.

Fig. 3. Construction of an interactive wPRF **wPRF** from a non-interactive wPRF **nl-wPRF**

where **setup** is a PPT algorithm, and **map** and **check** are deterministic polynomial time algorithms such that:

- **setup**(κ, ℓ, ϵ) outputs a string K of length polynomial in κ, ℓ and $1/\epsilon$.
- Completeness: For any polynomials $\ell, 1/\epsilon, \forall (x, w) \in R_{\ell(\kappa)}$,

$$\Pr_{\substack{K \leftarrow \text{setup}(\kappa, \ell(\kappa), \epsilon(\kappa)) \\ w^* \leftarrow \text{map}(K, x, w)}} [\text{check}(K, x, w^*) = 1] = 1.$$

- Pseudo-Uniqueness: For any polynomial $\ell, \forall x \in \{0, 1\}^{\ell(\kappa)}, \forall w_1, w_2$ such that $(x, w_1), (x, w_2) \in R_{\ell(\kappa)}$ (possibly $w_1 = w_2$),

$$\Pr_{\substack{K \leftarrow \text{setup}(\kappa, \ell(\kappa), \epsilon(\kappa)) \\ w_1^* \leftarrow \text{map}(K, x, w_1) \\ w_2^* \leftarrow \text{map}(K, x, w_2)}} [w_1^* \neq w_2^*] \leq \epsilon(\kappa)$$

- *Soundness*: For any polynomials $\ell, 1/\epsilon$, and any PPT adversary \mathcal{A} , $\text{Adv}_{\mathcal{A}}^{\text{CWM}}(\kappa)$ defined below is negligible:

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

We now present the construction below. The main building block of our construction is a statistically-sound NIWI (SNIWI) argument system NIWI. Before proceeding with the construction, let us try to solve a seemingly unrelated algorithmic problem, which we call the *pseudo-deterministic sampling* (PDS) problem over a polynomial-sized domain. We will later see how to use a solution for the PDS problem in our construction of p-UWM from CWM.

Pseudo-Deterministic Sampling for Small Domains. Let \mathcal{D} be an arbitrary distribution over some set X with a support of size n . Our goal is to design an algorithm PDS.sam that is polynomial-time in n and with only sampling access to \mathcal{D} , can *pseudo-deterministically* output an element from the support of \mathcal{D} . PDS.sam takes a reference string crs , and the pseudo-determinism is required to hold with high probability over the choice of crs . More formally, $(\text{PDS.setup}, \text{PDS.sam})$ is said to be a PDS scheme if:

1. $\text{PDS.setup}(n, \ell, \delta)$, with inputs the security parameter κ , a bound n on the support size of distributions over $\{0, 1\}^\ell$ that are to be handled, and a probability $\delta > 0$, outputs a common reference string crs of length polynomial in n, ℓ and $1/\delta$.
2. $\text{PDS.sam}^{\mathcal{D}}(\text{crs})$: Given an input crs and sampling access to a distribution \mathcal{D} over $\{0, 1\}^\ell$, the algorithm PDS.sam gets a polynomial number of samples from \mathcal{D} (polynomial in $|\text{crs}|$) and outputs an element in the support of \mathcal{D} , such that the following holds:
 - **Pseudo-determinism**: For all distributions \mathcal{D} over $\{0, 1\}^\ell$ with support size at most n ,

$$\Pr_{\substack{\text{crs} \leftarrow \text{PDS.setup}(n, \ell, \delta) \\ c_1 \leftarrow \text{PDS.sam}^{\mathcal{D}}(\text{crs}), c_2 \leftarrow \text{PDS.sam}^{\mathcal{D}}(\text{crs})}} [c_1 \neq c_2] \leq \delta$$

where the probability is over the choice of crs as well as the samples from \mathcal{D} .

We refer to an (n, ℓ, δ) -PDS as a PDS scheme setup with those parameters.

We now present a construction of a PDS scheme in [Figure 4](#).

Let \mathcal{D} be an arbitrary distribution over $\{0, 1\}^\ell$ with a support of size at most n .

1. PDS.setup(n, ℓ, δ) : Let $\mathcal{H} = \{H_\kappa\}_{\kappa \in \mathbb{N}}$ be a family of $2N$ -wise independent hash functions for $N = (2n/\delta)^3$. Sample a hash function $H \xleftarrow{\$} \mathcal{H}$, where $H : \{0, 1\}^{\ell + \log N} \rightarrow \{0, 1\}^\kappa$, κ is the security parameter. Set $\text{crs} = H$.
2. PDS.sam ^{\mathcal{D}} (crs) :
 - (a) Obtain a multi-set S of N samples from \mathcal{D} .
 - (b) Define the set $Y = \{x \mid i \mid i > 0 \text{ and } x \text{ has multiplicity at least } i \text{ in } S\}$.
 - (c) Let $\mathbf{H} = \{H(y) \mid y \in Y\}$, where each $h \in \mathbf{H}$ is a bit string of length κ .
 - (d) Return the lexicographically smallest element of \mathbf{H} .

Fig. 4. A pseudo-deterministic sampling algorithm for a small support

Lemma 3. ($\text{PDS.setup}, \text{PDS.sam}$) (*Figure 4*) is a PDS scheme.

Proof. Since PDS.sam satisfies the efficiency requirements, and always outputs an element in the support of \mathcal{D} , it remains to show that it satisfies the pseudo-determinism requirement.

Consider two independent runs of PDS.sam using the same H . Let S_i, Y_i denote the multi-set of samples and the set of count-appended samples in the two executions.

First, fix the sets Y_1 and Y_2 . Note that if the lexicographically smallest element in $\{H(y) \mid y \in Y_1 \cup Y_2\}$ is an element $H(y)$ for $y \in Y_1 \cap Y_2$, then PDS.sam outputs the same value in both runs. Now, since \mathcal{H} is $2N$ -wise independent, H behaves identical to a random function over $Y_1 \cup Y_2$. Hence, over the choice of H , the probability of error – i.e., that the outputs of PDS.sam are different – is upper bounded by $\frac{|Y_1 \Delta Y_2|}{|Y_1 \cup Y_2|} \leq \frac{|Y_1 \Delta Y_2|}{N}$.

Now we define the following “Good” event for the choice of (S_1, S_2) over the samples from \mathcal{D} (independent of H): For all x the support of \mathcal{D} , the multiplicity of x in S_1 and in S_2 are both in the range $[Np - N^{2/3}, Np + N^{2/3}]$, where p is the probability assigned to x by \mathcal{D} . When this condition holds, $|Y_1 \Delta Y_2| \leq n \cdot 2N^{2/3}$, where n is an upper bound on the size of the support of \mathcal{D} . Hence, conditioned on the Good event, the probability of error is at most $\frac{|n \cdot 2N^{2/3}|}{2N}$.

Finally, by [Lemma 1](#) and union bound (for each x in the support of \mathcal{D} , and each of Y_1, Y_2), the probability of the Good event not occurring is at most $2ne^{-N^{1/3}}$. Hence the probability of error is at most $2ne^{-N^{1/3}} + nN^{-1/3}$. Letting $N = (2n/\delta)^3$, this is at most δ . \square

Now we present the p-UWM scheme for an NP relation R . For simplicity, first we present a randomized map, and then point out how to derandomize it using the CRS.

1. The p-UWM setup outputs $\text{crs} = (\text{crs}_{\text{NIWI}}, \text{crs}_{\text{CWM}}, \text{crs}_{\text{PDS.sam}})$, which consists of the setup for a statistically sound NIWI proof system for the relation R , an α -CWM for the relation corresponding to the NIWI verifier, and a $(2^\alpha, \ell, \epsilon/2)$ -PDS scheme (as given above).
2. The p-UWM, on input $(x, w) \in R$, defines the distribution $\mathcal{D}_{x,w}$ as the distribution of $\text{map}(\text{NIWI}(x, w; \text{crs}_{\text{NIWI}}); \text{crs}_{\text{CWM}})$. Then it outputs $\text{PDS.sam}^{\mathcal{D}_{x,w}}$.
3. The p-UWM verifier is the same as `check`.

Completeness is easy to see. The soundness of this proof depends on the fact that for any false statement, there does not *exist* a NIWI proof that gets accepted (except with negligible probability, over the choice of crs_{NIWI}), and that CWM is (computationally) sound. For pseudo-uniqueness, first consider two runs of p-UWM (with the same setup) using the same witness w . In this case, from the compactness of CWM and the pseudo-determinism of the PDS scheme, the probability of the outputs differing is at most $\epsilon/2$. In the general case, when two different witnesses w_1, w_2 are used, we note that the distributions \mathcal{D}_{x,w_1} and \mathcal{D}_{x,w_2} are computationally indistinguishable from each other, thanks to the witness indistinguishability property of NIWI. Since PDS.sam is computationally efficient, this implies that the probability of the outputs differing given access to \mathcal{D}_{x,w_1} and \mathcal{D}_{x,w_2} (rather than two copies of \mathcal{D}_{x,w_1}) can only be negligibly more than $\epsilon/2$. Hence for any inverse polynomial ϵ , this error probability is bounded by ϵ , as required.

Finally, we address the fact the p-UWM mapping algorithm above was defined to be randomized, to carry out the implementation of (N samples from) $\mathcal{D}_{x,w}$, given (x, w) . Since the definition of pseudo-uniqueness involves only two runs of the mapping algorithm, it is enough to include a pairwise independent hash function in the CRS which would be used to derive the required amount of randomness as a function of its input (x, w) .

6 Mildly Compact WM implies NIZK

In this section, we show that a mildly compact WM - i.e., a CWM with compactness level $\alpha = \text{poly}(\kappa)$ for some fixed polynomial $\text{poly}(\cdot)$, independent of the statement size $|x|$ or the witness size $|w|$ implies the existence of NIZK argument system. As mentioned in the introduction, we generalize the recent work of [10] who constructed NIZKs from SNARG by replacing SNARG with CWM with the above compactness level. In particular, we show a construction of *hidden-bits generator with*

subset-dependent proofs (SDP-HBG) from α -CWM. This result, along with the compiler of [10] that transforms any NIZK in the hidden-bits model (HBM-NIZK) to NIZK in the CRS model using SDP-HBG implies a construction of NIZK from any α -CWM as long as $\alpha = \text{poly}(\kappa)$ as defined above. Following [10] we first provide the definition of SDP-HBG.

6.1 Hidden-Bits Generator with Subset-Dependent Proofs

Following [10], we recall the notion of hidden-bits generator with subset-dependent proofs (SDP-HBG).

Definition 8 (SDP-HBG). A hidden-bits generator with subset-dependent proofs (SDP-HBG) consists of four PPT algorithms ($\text{HBG}^{\text{sdp}}.\text{Setup}$, $\text{HBG}^{\text{sdp}}.\text{GenBits}$, $\text{HBG}^{\text{sdp}}.\text{Prove}$, $\text{HBG}^{\text{sdp}}.\text{Verify}$) defined as follows:

1. $\text{HBG}^{\text{sdp}}.\text{Setup}(1^\kappa, 1^n)$: The setup algorithm takes the security parameter 1^κ and the length parameter 1^n as input, and outputs a CRS crs .
2. $\text{HBG}^{\text{sdp}}.\text{GenBits}(\text{crs})$: The bits generation algorithm takes a CRS crs as input, and outputs a string $r \xleftarrow{\$} \{0, 1\}^n$ and a state st .
3. $\text{HBG}^{\text{sdp}}.\text{Prove}(\text{st}, I)$: The proving algorithm takes a state st and a subset $I \subseteq [n]$ as input, and outputs a proof π .
4. $\text{HBG}^{\text{sdp}}.\text{Verify}(\text{crs}, I, r_I, \pi)$: The verification algorithm takes a CRS crs , a subset $I \subseteq [n]$, a string $r_I \in \{0, 1\}^{|I|}$ and a proof π as input, and outputs either 1 or 0 indicating acceptance or rejection respectively.

A SDP-HBG is required to satisfy the following properties:

- **Correctness.** For any natural number n and $I \subseteq [n]$, we have:

$$\Pr \left[\begin{array}{l} \text{crs} \leftarrow \text{HBG}^{\text{sdp}}.\text{Setup}(1^\kappa, 1^n); \\ \text{HBG}^{\text{sdp}}.\text{Verify}(\text{crs}, I, r_I, \pi) = 1 : (r, \text{st}) \leftarrow \text{HBG}^{\text{sdp}}.\text{GenBits}(\text{crs}); \\ \pi \leftarrow \text{HBG}^{\text{sdp}}.\text{Prove}(\text{st}, I) \end{array} \right] = 1$$

- **Somewhat Computational Binding.** There exists a constant $\gamma < 1$ such that (1) for any polynomial $n = n(\kappa)$ and for all $\text{crs} \leftarrow \text{HBG}^{\text{sdp}}.\text{Setup}(1^\kappa, 1^n)$, there exists a subset $\mathcal{V}^{\text{crs}} \subseteq \{0, 1\}^n$ such that $|\mathcal{V}^{\text{crs}}| \leq 2^{n^\gamma \text{poly}(\kappa)}$ holds, and (2) for any PPT adversary \mathcal{A} , we have:

$$\Pr_{\text{crs} \leftarrow \text{HBG}^{\text{sdp}}.\text{Setup}(1^\kappa, 1^n)} \left[r_I \notin \mathcal{V}_I^{\text{crs}} \wedge \text{HBG}^{\text{sdp}}.\text{Verify}(\text{crs}, I, r_I, \pi) = 1 : (I, r_I, \pi) \leftarrow \mathcal{A}(\text{crs}) \right] = \text{negl}(\kappa).$$

where $\mathcal{V}_I^{\text{crs}} = \{r_I : r \in \mathcal{V}^{\text{crs}}\}$

- **Computational Hiding.** For any polynomial $n = n(\kappa)$, $I \subseteq [n]$, any PPT adversary \mathcal{A} , we have:

$$\left| \Pr [\mathcal{A}(\text{crs}, I, r_I, \pi, r_{\bar{I}}) = 1] - \Pr [\mathcal{A}(\text{crs}, I, r_I, \pi, r'_I) = 1] \right|$$

where r'_I denotes the substring of r on the positions corresponding to $\bar{I} = [n] \setminus I$.

A SDP-HBG is a weaker primitive than HBG, and [10] showed how to construct a SDP-HBG generically starting from any HBG.

6.2 Construction of SDP-HBG.

In this section, we show how to construct a SDP-HBG from CWM and OWF. Our construction requires the following ingredients:

- An λ -entropic leakage-resilient weak PRF (λ -LR-wPRF) $\mathcal{F} = \{F_K : \{0, 1\}^m \rightarrow \{0, 1\}\}_{K \in \{0, 1\}^k}$ (see Section 2.2.3), with key length $k = k(\kappa, \lambda) = \lambda \cdot \text{poly}(\kappa)$, input length $m = m(\kappa, \lambda) = \lambda \cdot \text{poly}(\kappa)$, and output length 1 bit. Here λ denotes the leakage parameter of the ELR-wPRF \mathcal{F} .
- A α -CWM $\text{cwm} = (\text{setup}, \text{map}, \text{check})$ for $\alpha = \text{poly}(\kappa)$ for an arbitrary polynomial $\text{poly}(\kappa)$ (independent of the length of the statement or witness) for the language L associated with the following relation R :

$$\left((k', \{x_i\}_{i \in [k']}, \{r_i\}_{i \in [k']}), K \right) \in R \iff r_i = F_K(x_i) \quad \forall i \in [k']$$

We now proceed to describe our construction.

1. $\text{HBG}^{\text{sdP}}.\text{Setup}(1^\kappa, 1^n)$: Do the following:
 - (a) Run $K \leftarrow \text{setup}(\kappa, \ell)$, where ℓ is the length of the statement mentioned in the relation above.
 - (b) For all $i \in [n]$, sample $x_i \xleftarrow{\$} \{0, 1\}^m$.
 Return $\text{crs} = (K, \{x_i\}_{i \in [n]})$.
2. $\text{HBG}^{\text{sdP}}.\text{GenBits}(\text{crs})$: Do the following:
 - (a) Parse the CRS as $\text{crs} = (K, \{x_i\}_{i \in [n]})$.
 - (b) Sample key $K \xleftarrow{\$} \{0, 1\}^k$, and compute $r_i = F_K(x_i)$ for all $i \in [n]$.
 Return $(r = \{r_i\}_{i \in [n]}, \text{st} = (\text{crs}, K, r))$.

3. $\text{HBG}^{\text{sdp}}.\text{Prove}(\text{st}, I) : \text{Parse } \text{st} = (\text{crs}, K, r) \text{ and } \text{crs} = ((K, \{x_i\}_{i \in [n]}).$
Then compute $w^* \leftarrow \text{map}(K, (|I|, x_I, r_I), K)$ and return $\pi := w^*$.
4. $\text{HBG}^{\text{sdp}}.\text{Verify}(\text{crs}, I, r_I, \pi) : \text{Parse } \text{crs} = ((K, \{x_i\}_{i \in [n]} \text{ and } \pi \text{ as } w^*.$
Return the output $\text{check}(K, (|I|, x_I, r_I), w^*)$.

Theorem 4. *Let κ be the security parameter. If there exist a λ -entropic leakage-resilient weak PRF and a α -CWM for all NP languages for $\alpha = \text{poly}(\kappa)$ (where $\text{poly}(\kappa)$ is some polynomial) such that $\alpha \leq \lambda$ and that satisfies adaptive soundness, then there exists an SDP-HBG that satisfies somewhat computational binding and computational hiding.*

The detailed proof of this theorem is presented in the full version of the paper. The intuition behind the proof is given in the technical overview.

7 UWM implies Deterministic-Prover NIZK

In this section we show that UWM implies *deterministic-prover* NIZK argument system (DP-NIZK) satisfying *perfect* zero-knowledge. Before this, we knew how to construct such a DP-NIZK argument system only from $i\mathcal{O}$. Below we briefly sketch the construction and defer to the full version of the paper for the details of the construction and proof.

A DP-NIZK argument system is NIZK argument system where the prover and verifier are both *deterministic*. Apart from completeness and soundness we require perfect zero-knowledge property to hold, i.e., the simulated proofs are identically distributed to the real proofs. The setup algorithm of our DP-NIZK chooses a random seed z for a length-doubling pseudorandom generator G and sets $y = G(z)$. The CRS crs of DP-NIZK consists of the CRS K of UWM and the value y . The prover of DP-NIZK on input some (x, w) in the relation runs the UWM prover to get a representative witness w^* for the statement \hat{x} stating that “either x is true or y is pseudorandom”, using w as the witness. Note that the prover is deterministic. In the proof of soundness, we sample y uniformly at random, so that y is not in the image of G , except with negligible probability. At this point the soundness of DP-NIZK follows from the soundness of UWM. To prove ZK, the simulator uses the witness z to simulate the proofs. The uniqueness property of UWM guarantees that proofs computed by either of the witnesses result in the same proof. Hence, the zero-knowledge property follows.

8 Perfectly Sound Verifier UWM implies $\mathbf{NP} = \mathbf{UP}$

In this section, we show that if a perfect sound verifier unique witness map (VUWM) exists (see Definition 6) then the complexity class \mathbf{NP} will be equal to the complexity class \mathbf{UP} , where \mathbf{UP} stands for unambiguous non-deterministic polynomial-time. Informally, the class \mathbf{UP} is the complexity class of decision problems solvable in polynomial time on an *unambiguous* Turing machine with at most *one* accepting path for each input. Hence it is easy to see that \mathbf{UP} contains the class \mathbf{P} and is contained in \mathbf{NP} . In the following we shall prove that $\mathbf{NP} \subseteq \mathbf{UP}$, assuming perfect sound VUWM. Let us first formally define the class \mathbf{UP} .

Definition 9 (Complexity class \mathbf{UP}). *A language $L \in \mathbf{UP}$ if there exists a two-input polynomial-time algorithm R and a constant c such that*

- *If $x \in L$, then there exists a unique certificate w with $|w| = O(|x|)^c$ such that $R(x, w) = 1$.*
- *If $x \notin L$, there is no certificate w with $|w| = O(|x|)^c$ such that $R(x, w) = 1$.*

Valiant and Vazirani [16] showed that $\mathbf{NP} \subseteq \mathbf{RP}^{\mathbf{promise-UP}}$, which means that there is a randomized reduction from any problem in \mathbf{NP} to a problem in $\mathbf{Promise-UP}$.

Theorem 5. *If perfectly-sound VUWM exists for an \mathbf{NP} relation R , then $L_R \in \mathbf{UP}$. In particular, if perfectly-sound VUWM exists for every \mathbf{NP} relation, then $\mathbf{NP} = \mathbf{UP}$.*

The proof of this theorem is presented in the full version of the paper.

References

1. Richard Beigel. On the relativized power of additional accepting paths. In *Proceedings: Fourth Annual Structure in Complexity Theory Conference, University of Oregon, Eugene, Oregon, USA, June 19-22, 1989*, pages 216–224. IEEE Computer Society, 1989.
2. Richard Beigel, Harry Buhrman, and Lance Fortnow. NP might not be as easy as detecting unique solutions. In Jeffrey Scott Vitter, editor, *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 203–208. ACM, 1998.
3. Elette Boyle, Gil Segev, and Daniel Wichs. Fully leakage-resilient signatures. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 89–108. Springer, Heidelberg, May 2011.
4. Suvradip Chakraborty, Manoj Prabhakaran, and Daniel Wichs. Witness maps and applications. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part I*, volume 12110 of *LNCS*, pages 220–246. Springer, Heidelberg, May 2020.

5. Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Cryptography against continuous memory attacks. In *51st FOCS*, pages 511–520. IEEE Computer Society Press, October 2010.
6. Eran Gat and Shafi Goldwasser. Probabilistic search algorithms with unique answers and their cryptographic applications. *Electron. Colloquium Comput. Complex.*, page 136, 2011.
7. Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *21st ACM STOC*, pages 25–32. ACM Press, May 1989.
8. Shafi Goldwasser, Ofer Grossman, and Dhiraj Holden. Pseudo-deterministic proofs. In Anna R. Karlin, editor, *ITCS 2018*, volume 94 of *LIPICs*, pages 17:1–17:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, January 2018.
9. Carmit Hazay, Adriana López-Alt, Hoeteck Wee, and Daniel Wichs. Leakage-resilient cryptography from minimal assumptions. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 160–176. Springer, Heidelberg, May 2013.
10. Fuyuki Kitagawa, Takahiro Matsuda, and Takashi Yamakawa. NIZK from SNARG. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 567–595. Springer, Heidelberg, November 2020.
11. Huijia Lin, Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation with non-trivial efficiency. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016, Part II*, volume 9615 of *LNCS*, pages 447–462. Springer, Heidelberg, March 2016.
12. Yanyi Liu and Rafael Pass. On one-way functions and kolmogorov complexity. In *61st FOCS*, pages 1243–1254. IEEE Computer Society Press, November 2020.
13. Charles Rackoff. Relativized questions involving probabilistic algorithms. *J. ACM*, 29(1):261–268, 1982.
14. Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th ACM STOC*, pages 475–484. ACM Press, May / June 2014.
15. Leslie G. Valiant. Relative complexity of checking and evaluating. *Inf. Process. Lett.*, 5(1):20–23, 1976.
16. Leslie G. Valiant and Vijay V. Vazirani. NP is as easy as detecting unique solutions. *Theor. Comput. Sci.*, 47(3):85–93, 1986.
17. Mark Zhandry. How to avoid obfuscation using witness PRFs. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part II*, volume 9563 of *LNCS*, pages 421–448. Springer, Heidelberg, January 2016.