Designated-verifier pseudorandom generators, and their applications

Geoffroy Couteau* and Dennis Hofheinz**

KIT, Karlsruhe

Abstract. We provide a generic construction of non-interactive zero-knowledge (NIZK) schemes. Our construction is a refinement of Dwork and Naor's (FOCS 2000) implementation of the hidden bits model using verifiable pseudorandom generators (VPRGs). Our refinement simplifies their construction and relaxes the necessary assumptions considerably.
As a result of this conceptual improvement, we obtain interesting new instantiations:

- A designated-verifier NIZK (with unbounded soundness) based on the computational Diffie-Hellman (CDH) problem. If a pairing is available, this NIZK becomes publicly verifiable. This constitutes the first fully secure CDH-based designated-verifier NIZKs (and more generally, the first fully secure designated-verifier NIZK from a non-generic assumption which does not already imply publicly-verifiable NIZKs), and it answers an open problem recently raised by Kim and Wu (CRYPTO 2018).
- A NIZK based on the learning with errors (LWE) assumption, and assuming a non-interactive witness-indistinguishable (NIWI) proof system for bounded distance decoding (BDD). This simplifies and improves upon a recent NIZK from LWE that assumes a NIZK for BDD (Rothblum et al., PKC 2019).

Keywords: non-interactive zero-knowledge, computational Diffie-Hellman, learning with errors, verifiable pseudorandom generators.

1 Introduction

Zero-knowledge proof systems allow a prover to convince someone of the truth of a statement, without revealing anything beyond the fact that the statement is true. After their introduction in the seminal work of Goldwasser, Micali, and Rackoff [20], they have proven to be a fundamental primitive in cryptography. Among them, *non-interactive zero-knowledge proofs* [5] (NIZK proofs), where the proof consists of a single flow from the prover to the verifier, are of particular interest, in part due to their tremendous number of applications in cryptographic primitives and protocols, and in part due to the theoretical and technical challenges that they represent.

* Supported by ERC grant "PREP-CRYPTO" (724307).
** Supported by ERC grant "PREP-CRYPTO" (724307) and DFG project GZ HO 4304/4-2.

On Building Non-Interactive Zero-Knowledge Proofs. It is known that zero-knowledge proofs for arbitrary NP languages can be constructed from any one-way function [19], and that this is a minimal assumption [30, 32, 39]. In contrast, non-interactive zero-knowledge proofs have proven to be considerably harder to construct. NIZKs in the plain model can only exist for trivial languages [31]; therefore, NIZKs for non-trivial languages are typically constructed in the *common reference string model*, where the prover and the verifier are given access to a common string honestly generated ahead of time in a setup phase. Generic constructions of NIZK proof systems for NP in the CRS model have been described from primitives such as doubly-enhanced trapdoor permutations [17], invariant signatures [21], and verifiable pseudorandom generators [16], where the last two are known to be also necessary for NIZKs. However, concrete instantiations of these primitives are currently known only from factorization-related assumption [5], pairing-based assumptions [8], and indistinguishability obfuscation [4, 9] (together with injective one-way functions). More recently, direct constructions of NIZKs in the CRS model have been given from pairings [24–26], or from strong and less-understood assumptions such as indistinguishability obfuscation [4, 38] and exponentially-strong KDM-secure encryption [7].

A fundamental and intriguing open question remains: is it possible to build NIZKs from other classical and well-established assumptions, such as discrete-logarithm-type assumptions, or lattice-based assumptions? Faced with the difficulty of tackling this hard problem upfront, the researchers have investigated indirect approaches, which can be divided into two main categories: the *bottom-up* approach, and the *top-down* approach.

The Bottom-Up Approach. This line of research fundamentally asks the following: starting from classical assumptions, either generic (OWF, public-key encryption) or concrete (CDH, LWE), how close to full-fledged NIZKs in the CRS model can we get, in terms of functionality? Early results in this direction have established the existence of NIZKs for NP in the *preprocessing model* (where the prover and the verifier execute ahead of time a preprocessing phase to generate respectively a secret proving key and a secret verification) assuming any one-way function [15], and *designated-verifier* NIZKs for NP (where anyone can compute a proof, but a secret verification key is required to verify a proof) from any semantically-secure public-key encryption scheme [33]. In addition to requiring the prover and/or the verifier to hold a secret key, these early results all suffered from a severe limitation: they only achieve a bounded form of soundness, where forging a proof for an incorrect statement is hard only if the prover is not given access to a verification oracle. This strongly limits their usability as a replacement for full-fledged NIZKs in most applications. More recently, various NIZK proof systems with unbounded soundness have been proposed, from the LPN assumption in the preprocessing model [6], from strong form of partially homomorphic encryption in the designated-verifier setting [11], and from LWE in the designated-prover setting [29] (where a secret key is required to compute a proof but anyone can verify a proof; the latter work also implies a NIZK with

unbounded soundness in the preprocessing model from a strong variant of the Diffie-Hellman assumption). A slightly different approach was taken in [2], where the authors introduce (and construct from the DDH assumption) implicit zero-knowledge proofs, which are not NIZKs, but can replace them in applications related to secure computation.

The Top-Down Approach. This line of research tackles the problem from another angle: sticking with the goal of building full-fledged NIZKs in the CRS model, it attempts to identify the minimal "missing piece" which would allow to build NIZKs from classical assumptions. The work of [34] conjectured that a NIZK proof system for a specific language (GapSVP) would allow to build a NIZK proof system for all of NP from lattice assumptions, and the work of [37] almost confirmed this conjecture, by establishing that a non-interactive zero-knowledge proof for a specific language (bounded distance decoding, BDD) would imply the existence of a full-fledged NIZK proof system for NP in the CRS model, from the LWE assumption.

1.1 Our Contribution

In this paper, we revisit the problem of building non-interactive zero-knowledge proofs for NP from classical assumptions, investigating both the bottom-up approach and the top-down approach.

Our starting point is a fresh view on the work [16] of Dwork and Naor. In a nutshell, they construct a NIZK proof for NP by implementing the hidden bits model (HBM, [17]) using a tool they call verifiable pseudorandom generator (VPRG).[1] Intuitively, a VPRG is a pseudorandom generator (PRG) that allows to selectively prove that certain parts of the PRG output are consistent (relative to a commitment to the PRG input).

In the first part of our work, we relax the definition of VPRGs, and show that the relaxed definition is still sufficient to implement the HBM (and thus to obtain NIZK proofs for NP). Unlike the definition of [16], our definition also generalizes to the designated-verifier setting. In the second part of our work, we show that our new definition allows for considerably simple and new instantiations, both in the designated-verifier and standard (publicly verifiable) setting. We obtain instantiations from computational assumptions which were so far not known to imply NIZKs for NP. Specifically, we provide:

- A designated-verifier NIZK (DVNIZK) system for NP from the CDH assumption (with adaptive unbounded soundness and adaptive multi-theorem zero-knowledge). If the underlying group allows for a (symmetric) pairing, our construction can be made publicly verifiable. This is the first DVNIZK for NP from a concrete (i.e., non-generic) assumption that is not already known to imply publicly verifiable NIZKs for NP. Our result resolves an open problem recently raised by Kim and Wu in [29], regarding the possibility of

[1] In the HBM, there exist unconditionally secure NIZK proofs [17].

building multi-theorem NIZKs from DDH in the preprocessing model. Note that our result achieves a strictly stronger form of NIZK and under a weaker assumption.
– A NIZK system for NP (satisfying adaptive soundness and adaptive multi-theorem zero-knowledge) that assumes LWE and a non-interactive witness-indistinguishable proof system Π' for BDD. If Π' is designated-verifier, resp. publicly verifiable, then so is our NIZK system for NP. Our scheme improves the mentioned work of [37] that requires non-interactive *zero-knowledge* proof system for BDD. (We comment below on what allows us to avoid the need for simulation inherent in the approach of [37].)

1.2 Our approach

The proof system of Dwork and Naor. To outline our conceptual contribution, we provide more background on the definitions and model of Dwork and Naor [16]. First, the hidden bits model (HBM) is an abstract model of computation for a prover and a verifier that allows to formulate the NIZK protocol for graph Hamiltonicity from [17] in a convenient way. In the HBM, the prover receives an ideally random string $\mathsf{hb} = (\mathsf{hb}_i)_{i=1}^t \in \{0,1\}^t$ of bits, as well as an NP-statement x with witness w. In order to prove x, the prover then selects a subset $S \subset [t]$ of bit indices and auxiliary information M. The verifier is then invoked with $\mathsf{hb}[S] = (\mathsf{hb}_i)_{i\in S}$ and M, and outputs 1 if it is convinced of the truth of x. [17] provide a NIZK proof in the HBM that is statistically sound and statistically zero-knowledge. (Of course, at least one of those properties will have to become computational when implementing the HBM.)

Now Dwork and Naor [16] implement the HBM using VPRGs. Formally, a VPRG is a pseudorandom generator $G : \{0,1\}^\lambda \to \{0,1\}^m$ which allows to construct commitments pvk to seeds (i.e., G-inputs) s and publicly verifiable openings of individual bits of $G(s)$ (relative to pvk). [16] require the following:

1. pvk information-theoretically determines a unique value y *in the image of* G,
2. valid openings to bits not consistent with the y determined by pvk *do not exist*, and
3. an opening computationally leaks nothing about unopened bits of y.

Given a VPRG, [16] implement the HBM as follows. The prover initially selects a seed $s \xleftarrow{\$} \{0,1\}^\lambda$ and then generates a commitment pvk to s. This implicitly sets $\mathsf{hb} = G(s)$. After selecting S, the prover then sends to the verifier pvk and an opening of $\mathsf{hb}[S]$.

Observe that this protocol may still allow the prover to cheat by choosing a "bad" seed s that might allow breaking soundness. However, since the HBM protocol of [17] is statistically sound, there can be only comparatively few "bad" HBM strings hb that allow cheating. Hence, the probability that there *exists* a seed s such that $\mathsf{hb} = G(s)$ is bad will be negligible.[2]

[2] A formal argument requires a little care in choosing parameters, and in randomizing hb with an additional component in the NIZK common reference string.

Our conceptual improvement. We show that points 1. and 2. from the VPRG definition can be simplified. Specifically:

- We require that pvk uniquely determines some y, but we do not require that y is in the image of G. Instead, we require that the bitlength $|\mathsf{pvk}|$ of pvk is short (i.e., independent of m). Observe that now up to $2^{|\mathsf{pvk}|}$ "bad" y (and thus "bad" hb) might exist. However, since $|\mathsf{pvk}|$ is still short (compared to m), essentially the original proof strategy of [17] applies.
- We only require that it is computationally infeasible to come up with an opening not consistent with y. This relaxation requires a careful tracking of "bad events" during the security proof, but is essentially compatible with the proof strategy from [16].

Our first change allows us to omit an explicit proof of consistency of pvk that was necessary in [16]. This simplification will be highly useful in our concrete instantiations. Furthermore, our second change allows to consider *designated-verifier* NIZKs. Indeed, observe that the original requirement 2. above states that no valid openings inconsistent with y exist. This excludes designated-verifier realizations of VPRGs in which the verifier secret key can be used to forge proofs. However, since most existing DVNIZK proofs have this property (otherwise, they could be made publicly verifiable by making the secret verification key public), they are not helpful to construct VPRGs in the sense of [16]. In contrast, our relaxation is compatible with existing DVNIZKs (and indeed our first VPRG instantiation crucially relies on DVNIZKs).

Concrete constructions. We offer two VPRG constructions from concrete assumptions. The first construction assumes a CDH group $\mathbb{G} = \langle g \rangle$ of (not necessarily prime) order n. A seed is an exponent $s \in \mathbb{Z}_n$, and a commitment to s is g^s. Given public $u_i, v_i \in \mathbb{G}$ (for $i \in [t]$), the i-th bit $G(s)_i$ of the PRG image is $B(u_i^s, v_i^s)$, where B is a hard-core predicate of the CDH function. A proof π_i that certifies a given $G(s)_i$ consists of u_i^s, v_i^s, as well as proofs that both (g, g^s, u_i, u_i^s) and (g, g^s, v_i, v_i^s) are Diffie-Hellman tuples. In a designated-verifier setting, such proofs are known from hash proof systems [10,13]. Alternatively, a symmetric pairing $\mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ can be used to check the Diffie-Hellman property of these tuples even without explicit proof.

Our second construction assumes LWE and uses the notion of homomorphic commitments from [22]. These commitments have a "dual-mode" flavor, much like the commitments from [14,26]. Specifically, under LWE, the public parameters of these commitments can be switched between a "binding" mode (in which commitments are perfectly binding) and a "hiding" mode (in which commitments are statistically hiding). Furthermore, given a commitment com_s to s, it is possible to publicly compute a commitment $\mathsf{com}_{C(s)}$ to $C(s)$ for any (a-priori bounded) circuit C.

In our construction, we will assume any PRG G, and set com_s to be a commitment to a PRG seed $s \in \{0,1\}^\lambda$. Let G_i be a circuit that computes the i-th bit of G. An opening of the i-th bit is then an opening of the commitment $\mathsf{com}_{G_i(s)}$ to $G_i(s)$. (Note that $\mathsf{com}_{G_i(s)}$ can be publicly computed from com_s.)

Unfortunately, in the construction of [22], the opening of commitments may reveal sensitive information about intermediate computation results (or even about s in our case).

Hence, we will have to assume an additional proof system to open commitments without revealing additional information. For the commitments of [22], the corresponding language is the language of a BDD problem. Fortunately, the strong secrecy properties of these commitments allow us to restrict ourselves to a witness-indistinguishable (and not necessarily zero-knowledge) proof system for BDD.[3]

Relation to [37]. We note that recently, [37] established a reduction from NIZKs for NP from LWE to the existence of an *NIZK* for BDD, also through implementing the HBM. Informally, and casting the construction of their work in the language of VPRGs,[4] the core reason why a NIZK was required in [37] is the need for a consistency proof for pvk (as in [16]). Since the consistency of pvk is a unique-witness relation, and since the proof must hide predicates of the seed, it does not seem feasible to replace this NIZK, e.g., by a NIWI or a witness-hiding proof. We note that although NIWIs for NP imply the existence of NIZKs for NP, it is not clear whether a NIWI for a simple language such as BDD can be used to build a NIZK for BDD.

Relation to [1]. We also note that Abusalah [1] also implements the HBM using Diffie-Hellman-related assumptions (such as CDH in a pairing-friendly group). However, he does not follow the PRG-based paradigm of [16] that we refine. Instead, he directly generalizes the original HBM implementation of [17] to generalizations of trapdoor permutations.

1.3 Concurrent Works

Concurrently and independently to our work, two other works [27, 35] have achieved a result comparable to the first of our two main contributions, namely, designated-verifier non-interactive zero-knowledge proofs for NP from CDH. In all three works, the construction proceeds in a comparable way, by designing a CDH-based primitive which allows to compile the hidden-bit model into a designated-verifier NIZK. We summarize below the main differences between our works.

[3] One might wonder why we do not follow another route to obtain VPRGs from NIWI proofs for BDD. Specifically, [3, 23] construct even verifiable random *functions* from a NIWI for a (complex) LWE-related language. However, these constructions inherently use disjunctions, and it seems unlikely that the corresponding NIWIs can be reduced to the BDD language.

[4] The actual construction of [37] relies on a new notion of public-key encryption with prover-assisted oblivious ciphertext sampling, but the high level idea is comparable to the VPRG-based approach. A side contribution of our construction is that it is conceptually much simpler and straightforward than the construction of [37].

- The work of [35] provides in addition a construction of *malicious* designated-verifier NIZK for NP, where the setup consists of an (honestly generated) common random string and the verifier then gets to choose his own (potentially malicious) public/secret key pair to generate and verify proofs. The assumption underlying their construction is a stronger "one-more type" variant of CDH (i.e., the hardness of solving $n+1$ CDH challenges given n calls to an oracle solving CDH).
- The work of [27] provides two relatively different additional constructions of NIZKs: a *designated-prover* NIZK for NP with proofs of size $|C| + \mathsf{poly}(\lambda)$ (where C is the circuit checking the NP relation), under a strong Diffie-Hellman-type assumption over pairing groups, and a *preprocessing NIZK* for NP with proofs of size $|C| + \mathsf{poly}(\lambda)$ from the DDH assumption over pairing-free groups.
- The construction of NIZKs for NP assuming LWE and a NIWI for BDD is new to our work.

1.4 Organization

Section 2 introduces necessary preliminaries about non-interactive proof systems. Section 3 formally introduces designated-verifier pseudorandom generators, and defines their security properties. Section 4 provides a generic construction of a (designated-verifier) non-interactive zero-knowledge proof system for NP from our relaxed and generalized notion of DVPRGs, by instantiating the hidden bit model. Section 5 provides two instantiations of DVPRGs, from the CDH assumption in arbitrary group, and from the LWE assumption assuming in addition a NIWI proof system for BDD (where the resulting scheme is publicly verifiable iff the NIWI scheme is publicly verifiable).

2 Preliminaries

Notation. Throughout this paper, λ denotes the security parameter. A probabilistic polynomial time algorithm (PPT, also denoted *efficient* algorithm) runs in time polynomial in the (implicit) security parameter λ. A function f is *negligible* if for any positive polynomial p there exists a bound $B > 0$ such that, for any integer $k \geq B$, $|f(k)| \leq 1/|p(k)|$. An event occurs with *overwhelming probability* when its probability is at least $1 - \mathsf{negl}(\lambda)$ for a negligible function negl. Given a finite set S, the notation $x \xleftarrow{\$} S$ means a uniformly random assignment of an element of S to the variable x. We represent adversaries as interactive probabilistic Turing machines; the notation $\mathsf{Adv}^{\mathcal{O}}$ indicates that the machine Adv is given oracle access to $\mathcal{O}$. Adversaries will sometimes output an arbitrary state st to capture stateful interactions. For an integer n, $[n]$ denotes the set of integers from 1 to n. Given a string x of length n, we denote by x_i its ith bit (for any $i \leq n$), and by $x[S]$ the subsequence of the bits of x indexed by a subset S of $[n]$.

2.1 Non-Interactive Zero-Knowledge

We recall the definition of non-interactive zero-knowledge (NIZK) proofs and argument.

Definition 1 (Non-Interactive Zero-Knowledge Argument System). *A non-interactive zero-knowledge argument system for an* NP*-language* $\mathscr{L}$ *with relation* $R_{\mathscr{L}}$ *is a triple of probabilistic polynomial-time algorithms* (Setup, Prove, Verify) *such that*

- $\mathsf{Setup}(1^\lambda)$*, outputs a common reference string* crs *and a trapdoor* $\mathcal{T}$*,*
- $\mathsf{Prove}(\mathsf{crs}, x, w)$*, on input the crs* crs*, a word* x*, and a witness* w*, outputs a proof* $\boldsymbol{\pi}$*,*
- $\mathsf{Verify}(\mathsf{crs}, x, \pi, \mathcal{T})$*, on input the crs* crs*, a word* x*, a proof* $\boldsymbol{\pi}$*, and the trapdoor* $\mathcal{T}$*, outputs* $b \in \{0,1\}$*,*

which satisfies the completeness, soundness, and zero-knowledge properties defined below.

If the trapdoor $\mathcal{T}$ of the non-interactive proof system is set to $\perp$ (or, alternatively, if it is included in the crs), we call the argument system *publicly verifiable*. Otherwise, we call it a *designated-verifier non-interactive argument system*. If the soundness guarantee holds with respect to computationally unbounded adversary, we have a NIZK *proof* system.

Definition 2 (Perfect Completeness). *A non-interactive argument system* (Setup, Prove, Verify) *for an* NP*-language* $\mathscr{L}$ *with witness relation* $R_{\mathscr{L}}$ *satisfies perfect completeness if for every* $x \in \mathscr{L}$ *and every witness* w *such that* $R_{\mathscr{L}}(x, w) = 1$*,*

$$\Pr[(\mathsf{crs}, \mathcal{T}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda), \boldsymbol{\pi} \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w) : \mathsf{Verify}(\mathsf{crs}, x, \boldsymbol{\pi}, \mathcal{T}) = 1] = 1.$$

The soundness notion can come in several flavors: it is *non-adaptive* if the adversary must decide on a word on which to forge a proof before the common reference string is drawn, and it is *adaptive* if the adversary can dynamically choose the word given the common reference string. We will consider a strong variant of adaptive soundness, denoted *unbounded adaptive soundness*, where the adversary is given oracle access to a verification oracle. Note that in the publicly-verifiable setting, this is equivalent to the standard soundness notion, where the adversary must forge a valid proof on an incorrect statement without the help of any oracle. However, in the designated-verifier setting, this is a strictly stronger notion: the standard soundness notion only guarantees, in this setting, that the argument system remains sound as long as the prover receives at most logarithmically many feedback on previous proofs. On the other hand, if the argument system satisfies unbounded soundness, its soundness is maintained even if the adversary receives an arbitrary (polynomial) number of feedback on previous proofs.

Definition 3 (Unbounded Adaptive Soundness). *A non-interactive argument system* $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ *for an* NP*-language* $\mathscr{L}$ *with relation* $R_{\mathscr{L}}$ *satisfies unbounded adaptive soundness if for any PPT* $\mathcal{A}$*,*

$$\Pr\left[\begin{array}{l}(\mathsf{crs}, \mathcal{T}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda), \\ (x, \boldsymbol{\pi}) \xleftarrow{\$} \mathcal{A}^{\mathcal{O}(\mathsf{crs}, \cdot, \cdot, \mathcal{T})}(\mathsf{crs}) \end{array} : \mathsf{Verify}(\mathsf{crs}, x, \boldsymbol{\pi}, \mathcal{T}) = 1 \wedge x \notin \mathscr{L}\right] \approx 0,$$

where $\mathcal{A}$ *can make polynomially many queries to an oracle* $\mathcal{O}(\mathsf{crs}, \cdot, \cdot, \mathcal{T})$ *which, on input* $(x, \boldsymbol{\pi})$*, outputs* $\mathsf{Verify}(\mathsf{crs}, x, \boldsymbol{\pi}, \mathcal{T})$*.*

We now define zero-knowledge, which can again come in several flavors. We will consider *adaptive* zero-knowledge argument systems, where the adversary is allowed to pick a word on which to forge a proof after seeing the common reference string. We will also distinguish *single-theorem* zero-knowledge, in which the prover generates a single proof (and the length of the common reference string can be larger than the length of the statement to prove) and *multi-theorem* zero-knowledge (where the adversary can adaptively ask for polynomially many proofs on arbitrary pairs (x, w) for the same common reference string).

Definition 4 (Adaptive Single-Theorem Zero-Knowledge). *A non-interactive argument system* $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ *for an* NP*-language* $\mathscr{L}$ *with relation* $R_{\mathscr{L}}$ *satisfies (adaptive)* single-theorem zero-knowledge *if for any stateful PPT algorithm* $\mathcal{A}$*, there exists a simulator* $(\mathsf{Sim}_0, \mathsf{Sim}_1)$ *such that*

$$\left|\Pr\left[\begin{array}{l}(\mathsf{crs}, \mathcal{T}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda), \\ (x, w) \xleftarrow{\$} \mathcal{A}(\mathsf{crs}, \mathcal{T}), \\ \pi \xleftarrow{\$} \mathsf{Prove}(\mathsf{crs}, x, w)\end{array} : (R_{\mathscr{L}}(x, w) = 1) \wedge (\mathcal{A}(\pi) = 1)\right] - \right.$$
$$\left.\Pr\left[\begin{array}{l}(\mathsf{crs}, \mathcal{T}) \xleftarrow{\$} \mathsf{Sim}_0(1^\lambda), \\ (x, w) \xleftarrow{\$} \mathcal{A}(\mathsf{crs}, \mathcal{T}), \\ \pi \xleftarrow{\$} \mathsf{Sim}_1(\mathsf{crs}, \mathcal{T}, x)\end{array} : (R_{\mathscr{L}}(x, w) = 1) \wedge (\mathcal{A}(\pi) = 1)\right]\right| \approx 0.$$

Definition 5 (Adaptive Multi-Theorem Zero-Knowledge). *A non-interactive argument system* $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ *for an* NP*-language* $\mathscr{L}$ *with relation* $R_{\mathscr{L}}$ *satisfies (adaptive)* multi-theorem zero-knowledge *if for any stateful PPT algorithm* $\mathcal{A}$*, there exists a simulator* $(\mathsf{Sim}_0, \mathsf{Sim}_1)$ *such that* $\mathcal{A}$ *has negligible advantage in distinguishing the experiments* $\mathsf{Exp}^{zk,0}_{\mathcal{A}}(1^\lambda)$ *and* $\mathsf{Exp}^{zk,1}_{\mathcal{A}}(1^\lambda)$ *given on Figure 1.*

Note that $\mathcal{O}_{\mathsf{sim}}$ is only given the witness w to artificially enforce that $\mathcal{A}$ queries only words x in the language $\mathscr{L}$.

Zero-knowledge is a strong, simulation-style security notion. A common relaxation of zero-knowledge to an indistinguishability-based security notion is known as *witness-indistinguishability*.

Definition 6 (Computational Witness-Indistinguishability). *A non-interactive proof system* $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ *for an* NP*-language* $\mathscr{L}$ *with relation* $R_{\mathscr{L}}$

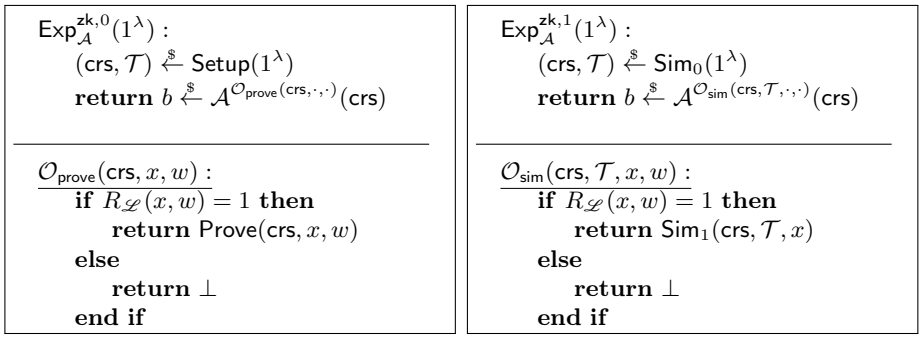

Fig. 1. Experiments $\mathsf{Exp}^{\mathsf{zk},0}_{\mathcal{A}}(1^\lambda)$ and $\mathsf{Exp}^{\mathsf{zk},1}_{\mathcal{A}}(1^\lambda)$, and oracles $\mathcal{O}_{\mathsf{prove}}(\mathsf{crs}, x, w)$ and $\mathcal{O}_{\mathsf{sim}}(\mathsf{crs}, \mathcal{T}, x, w)$, for the (adaptive) multi-theorem zero-knowledge property of a non-interactive argument system. $\mathcal{A}$ outputs $b \in \{0, 1\}$.

is (computationally) witness-indistinguishable *if for any PPT algorithm* $\mathcal{A}$*,*

$$\left| \Pr\left[\begin{array}{ll} (\mathsf{crs}, \mathcal{T}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda), & \mathcal{A}(\mathsf{crs}, \boldsymbol{\pi}) = 1 \\ (x, w_0, w_1) \xleftarrow{\$} \mathcal{A}(\mathsf{crs}), & : \wedge R_{\mathscr{L}}(x, w_0) = 1 \\ \boldsymbol{\pi} \xleftarrow{\$} \mathsf{Prove}(\mathsf{crs}, x, w_0) & \wedge R_{\mathscr{L}}(x, w_1) = 1 \end{array}\right] - \Pr\left[\begin{array}{ll} (\mathsf{crs}, \mathcal{T}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda), & \mathcal{A}(\mathsf{crs}, \boldsymbol{\pi}) = 1 \\ (x, w_0, w_1) \xleftarrow{\$} \mathcal{A}(\mathsf{crs}), & : \wedge R_{\mathscr{L}}(x, w_0) = 1 \\ \boldsymbol{\pi} \xleftarrow{\$} \mathsf{Prove}(\mathsf{crs}, x, w_1) & \wedge R_{\mathscr{L}}(x, w_1) = 1 \end{array}\right] \right| \approx 0$$

We call such a proof system a non-interactive witness-indistinguishable (NIWI) proof system*.*

It is known that the existence of a NIWI proof system for NP implies the existence of a NIZK proof system for NP in the CRS model [17]. However, this does not extend to proof systems for specific languages: the existence of a NIWI proof system for a language $\mathscr{L}$ does not generally imply the existence of a NIZK proof system in the CRS model for the same language.

3 Designated-Verifier Pseudorandom Generators

Verifiable pseudorandom generators (VPRG) have been introduced in the seminal paper of Dwork and Naor [16], as a tool to construct non-interactive witness-indistinguishable proofs and NIZKs in the CRS model. Informally, a VPRG enhances a PRG with verifiability properties: the prover can compute a kind commitment to the seed (called the *verification key*), and issue proofs that a given position i of the pseudorandom string stretched from the committed seed is equal to a given bit. Furthermore, this proof does not leak anything about the output values at positions $j \neq i$.

In this section, we revisit the notion of verifiable pseudorandom generators. Toward our goal of building VPRGs from new assumptions, we significantly weaken the binding property of VPRGs (which states, informally, that the verification key binds the prover to the seed) to a security notion that is simpler to achieve and still allows to build NIZKs in the CRS model, and we extend the definition to the more general setting of *designated-verifier* VPRGs (DVPRGs) (this strictly encompasses public VPRGs since we recover the standard notion by restricting the secret verification key to be $\perp$).

3.1 On Defining DVPRGs

A natural attempt to define the binding property of a DVPRG would be as follows: it should be infeasible, for any polytime adversary, to output two accepting proofs π_0 and π_1 that a given output of the PRG is equal to 0 and 1 respectively (relative to the same committed parameters) . However, this security notion turns out to be too weak for the construction of non-interactive witness-indistinguishable proofs from VPRG of [16]. Intuitively, this stems from the fact that a cheating prover will never send more than a single proof for a given output, hence we cannot extract two contradictory proofs from this adversary. Instead, the argument of [16] crucially rely on the following stronger definition: a VPRG is binding if for every (possibly malicious) public verification key pvk, there exists a single associated string x in the range of the stretching algorithm of the DVPRG, and for any accepting proof π of correct opening to a subset $y[I]$ of the bits of a string y, it must hold that $y[I] = x[I]$.

Unfortunately, this binding property turns out to be too strong for our purpose. The reason is that we seek to build candidate DVPRGs from assumptions such as LWE, where natural approaches lead to schemes where there exists malicious public verification keys associated to strings which are *not* in the range of the DVPRG, and which cannot be distinguished from honest verification keys (typically, in our LWE-based construction, an honest verification key will be a list of LWE samples, which are indistinguishable from random samples). A comparable issue arose in the work of [37], which tackled this issue by appending to the verification key (or, in their language, the public key of an obliviously-sampleable encryption scheme) a NIZK proof of validity.

Instead, we opt for a different approach and introduce a weaker binding property for DVPRGs, which does not require assuming any specific structure of the public verification key *beyond its length*. Namely, we consider the following notion: a DVPRG is binding if there exists a (possibly inefficient) extractor Ext such that no PPT adversary can output a triple (pvk, i, π) where π is a proof of correct opening of position i to $1 - x_i$, and $x = \mathsf{Ext}(\mathsf{pvk})$. Note that our definition does only consider verification keys generated by a computationally bounded adversary (instead of arbitrary pvk), and does not require pvk to be in the range of the DVPRG. This binding notion would in fact be trivial to achieve without further constraints (e.g. one could define pvk to be a sequence of extractable commitments to each bits of the pseudorandom string stretched from the seed), hence we further require that pvk must be short (of size $s(\lambda)$, for

a polynomial s independent of the stretch of the DVPRG). Afterward, we prove that this weaker notion still suffices to build NIZKs for NP in the CRS model.

Generalizing to the designated-verifier setting, where verification can involve a secret-verification key, we strengthen the above property to the *unbounded binding* property, which states that no PPT adversary can produce a triple (pvk, i, π) as above, even given oracle access to a verification oracle (which has the secret verification key hardcoded). We note that the above weakening of the binding notion is also necessary for our generalization to the designated-verifier setting: in this setting, the stronger binding notion of [16] does typically *not* hold, since there always exists accepting proofs of opening to an incorrect bit (if this was not the case, we could safely make the secret verification key public, since it would not allow to find proofs of opening to incorrect values); however, it is infeasible to *find* such proof (without knowing the secret verification key). Below, we formally introduce designated-verifier pseudorandom generators and the corresponding security notions.

3.2 Definition

Definition 7 (Designated-Verifier Pseudorandom Generator). *A designated-verifier pseudorandom generator (*DVPRG*) is a four-tuple of efficient algorithms* $(\mathsf{Setup}, \mathsf{Stretch}, \mathsf{Prove}, \mathsf{Verify})$ *such that*

- $\mathsf{Setup}(1^\lambda, m)$*, on input the security parameter (in unary) and a bound* $m(\lambda) = \mathsf{poly}(\lambda)$*, outputs a pair* $(\mathsf{pp}, \mathcal{T})$ *where* pp *is a set of public parameters (which contains* 1^λ*), and* $\mathcal{T}$ *is a trapdoor;*
- $\mathsf{Stretch}(\mathsf{pp})$*, on input the public parameters, outputs a triple* $(\mathsf{pvk}, x, \mathsf{aux})$*, where* pvk *is a public verification key of polynomial length* $s(\lambda)$ *independent of* m*,* x *is an* m*-bit pseudorandom string, and* aux *is an auxiliary information;*
- $\mathsf{Prove}(\mathsf{pp}, \mathsf{aux}, i)$*, on input the public parameters, auxiliary informations* aux*, an index* $i \in [m]$*, outputs a proof* π*;*
- $\mathsf{Verify}(\mathsf{pp}, \mathsf{pvk}, \mathcal{T}, i, b, \pi)$*, on input the public parameters, a public verification key* pvk*, a trapdoor* $\mathcal{T}$*, a position* $i \in [m]$*, a bit* b*, and a proof* π*, outputs a bit* β*;*

which is in addition complete*,* hiding*, and* binding*, as defined below.*

Note that the above definition also captures publicly verifiable pseudorandom generators, which are DVPRGs where we restrict $\mathsf{Setup}(1^\lambda, m)$ to always output pairs of the form $(\mathsf{pp}, \perp)$.

Definition 8 (Completeness of a DVPRG**).** *For any* $i \in [m]$*, a* perfectly complete DVPRG *scheme* $(\mathsf{Setup}, \mathsf{Stretch}, \mathsf{Prove}, \mathsf{Verify})$ *satisfies:*

$$\Pr\left[\begin{array}{l}(\mathsf{pp}, \mathcal{T}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, m),\\ (\mathsf{pvk}, x, \mathsf{aux}) \xleftarrow{\$} \mathsf{Stretch}(\mathsf{pp}), : \mathsf{Verify}(\mathsf{pp}, \mathsf{pvk}, \mathcal{T}, i, x_i, \pi) = 1\\ \pi \xleftarrow{\$} \mathsf{Prove}(\mathsf{pp}, \mathsf{aux}, i),\end{array}\right] = 1.$$

We now define the binding property of a DVPRG. We consider a flavor of the binding property which is significantly weaker than the one considered in [16], yet still suffices for the application to NIZKs (see the discussion in Section 3.1).

Definition 9 (Binding Property of a DVPRG). *Let* (Setup, Stretch, Prove, Verify) *be a* DVPRG. *A* DVPRG *is* binding *if there exists a (possibly inefficient)* extractor Ext *such that for any PPT* $\mathcal{A}$*, it holds that*

$$\Pr\left[\begin{array}{l}(\mathsf{pp},\mathcal{T}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, m),\\ (\mathsf{pvk}, i, \pi) \xleftarrow{\$} \mathcal{A}(\mathsf{pp}), \qquad : \mathsf{Verify}(\mathsf{pp},\mathsf{pvk},\mathcal{T}, i, 1-x_i, \pi) = 1\\ x \leftarrow \mathsf{Ext}(\mathsf{pvk})\end{array}\right] \approx 0.$$

As for non-interactive zero-knowledge proofs, the designated-verifier setting requires to explicitly consider whether the adversary is given access to a verification oracle. We therefore extend the above definition and consider the *unbounded* binding property:

Definition 10 (Unbounded Binding Property of a DVPRG). *Let* (Setup, Stretch, Prove, Verify) *be a* DVPRG. *A* DVPRG *satisfies* unbounded binding *if there exists a (possibly inefficient)* extractor Ext *such that for any PPT* $\mathcal{A}$*, it holds that*

$$\Pr\left[\begin{array}{l}(\mathsf{pp},\mathcal{T}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, m),\\ (\mathsf{pvk}, i, \pi) \xleftarrow{\$} \mathcal{A}^{\mathsf{Verify}(\mathsf{pp},\cdot,\mathcal{T},\cdot,\cdot,\cdot)}(\mathsf{pp}), : \mathsf{Verify}(\mathsf{pp},\mathsf{pvk},\mathcal{T}, i, 1-x_i, \pi) = 1\\ x \leftarrow \mathsf{Ext}(\mathsf{pvk})\end{array}\right] \approx 0.$$

Note that in the case of publicly verifiable pseudorandom generators, where $\mathcal{T}$ *is set to* $\perp$*, this security notion is equivalent to the binding property.*

We now define equivocability. Intuitively, it states that no computationally bounded adversary can distinguish honestly generated proofs of correctness for bits of the pseudorandom sequence from simulated proofs (using $\mathcal{T}$) of opening to true random bits.

Definition 11 (Equivocability of a DVPRG). *A designated-verifier pseudorandom generator* (Setup, Stretch, Prove, Equivocate, Verify) *is* equivocable *if there are two additional algorithms* (SimSetup, Equivocate) *such that*

- $\mathsf{SimSetup}(1^\lambda, m)$*, on input the security parameter in unary, outputs a triple* $(\mathsf{pp}, \mathcal{T}, \mathcal{T}_s)$*,*
- $\mathsf{Equivocate}(\mathsf{pp},\mathsf{pvk}, i, b, \mathcal{T}_s)$*, on input the public parameters, a public verification key* pvk*, an index* $i \in [m]$*, a bit* b*, and a simulation trapdoor* $\mathcal{T}_s$*, outputs a simulated proof* π'*;*

such that the following distributions are computationally indistinguishable:

$$\left\{\begin{array}{l}(\mathsf{pp},\mathcal{T}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, m),\\ (\mathsf{pvk}, x, \mathsf{aux}) \xleftarrow{\$} \mathsf{Stretch}(\mathsf{pp}) \\ \boldsymbol{\pi} \xleftarrow{\$} (\mathsf{Prove}(\mathsf{pp}, \mathsf{aux}, i))_i\end{array} : (\mathsf{pp}, \mathsf{pvk}, \mathcal{T}, x, \boldsymbol{\pi})\right\} = D_0$$

$$\approx \left\{\begin{array}{l}(\mathsf{pp},\mathcal{T},\mathcal{T}_s) \xleftarrow{\$} \mathsf{SimSetup}(1^\lambda, m),\\ (\mathsf{pvk}, x', \mathsf{aux}) \xleftarrow{\$} \mathsf{Stretch}(\mathsf{pp}), x \xleftarrow{\$} \{0,1\}^m, \\ \boldsymbol{\pi} \xleftarrow{\$} (\mathsf{Equivocate}(\mathsf{pp}, \mathsf{pvk}, i, x_i, \mathcal{T}_s))_i\end{array} : (\mathsf{pp}, \mathsf{pvk}, \mathcal{T}, x, \boldsymbol{\pi})\right\} = D_1.$$

A weaker variant of equivocability is the following *hiding* property, which states that an adversary cannot guess the value of a particular output (with non-negligible advantage over the random guess), even if he is given the values of all other outputs together with proofs of correct opening. This notion is implied by the equivocability property, and it suffices for for the Dwork and Naor construction of a NIZK proof system for NP; however, equivocable DVPRGs allow for a simpler and more direct construction of NIZKs, without having to rely on the FLS transform which constructs NIZKs from NIWI [17].

Definition 12 (Hiding Property of a DVPRG). *A* DVPRG *scheme* (Setup, Stretch, Prove, Verify) *is* hiding *if for any $i \in [m]$ and any PPT adversary $\mathcal{A}$ that outputs bits, it holds that:*

$$\Pr\left[\begin{array}{l}(\mathsf{pp},\mathcal{T}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, m),\\ (\mathsf{pvk}, x, \mathsf{aux}) \xleftarrow{\$} \mathsf{Stretch}(\mathsf{pp}), : \mathcal{A}(\mathsf{pp}, \mathsf{pvk}, i, (x_j, \pi_j)_{j\neq i}) = x_i\\ (\pi_j \xleftarrow{\$} \mathsf{Prove}(\mathsf{pp}, \mathsf{aux}, j))_j\end{array}\right] \approx 1/2.$$

Eventually, we define an additional security notion, the *consistency*, which will prove useful to analyze the unbounded binding property of one of our candidates:

Definition 13 (Consistency of a DVPRG). *Given a* DVPRG (Setup, Stretch, Prove, Verify) *and a pair $(\mathsf{pp},\mathcal{T}) = \mathsf{Setup}(1^\lambda, m; r)$ for some random coin r, we define for any ε the set ε-Good(r) to be the set of 4-tuples $(\mathsf{pvk}, i, \pi, x_i)$ satisfying*

$$\Pr\left[\mathcal{T}' \xleftarrow{\$} \mathtt{Dist}(r) : \mathsf{Verify}(\mathsf{pp}, \mathsf{pvk}, \mathcal{T}', i, x_i, \pi) = 1\right] \geq \varepsilon,$$

where $\mathtt{Dist}(r)$ samples random pairs $(\mathsf{pp}', \mathcal{T}')$ with $\mathsf{Setup}(1^\lambda, m)$ subject to the constraint $\mathsf{pp}' = \mathsf{pp}$, and outputs $\mathcal{T}'$. Note that for any $\varepsilon' \geq \varepsilon$, it holds that ε'-Good$(r) \subset \varepsilon$-Good(r). Then, we say that a DVPRG *is* consistent *if there exists a negligible function ε such that for any PPT adversary $\mathcal{A}$,*

$$\Pr\left[\begin{array}{l}r \xleftarrow{\$} R, (\mathsf{pp},\mathcal{T}) \leftarrow \mathsf{Setup}(1^\lambda, m; r), (\mathsf{pvk}, i, \pi, b) \xleftarrow{\$} \mathcal{A}(\mathsf{pp}) :\\ (\mathsf{pvk}, i, \pi, b) \in \varepsilon\text{-Good}(r) \setminus 1\text{-Good}(r)\end{array}\right] \approx 0.$$

In the full version of this paper [12], we prove the following:

Theorem 14. *Let $\mathcal{G} = (\mathsf{Setup}, \mathsf{Stretch}, \mathsf{Prove}, \mathsf{Verify})$ be a binding and consistent* DVPRG*, such that for any r, the distribution $\mathtt{Dist}(r)$ is efficiently sampleable. Then $\mathcal{G}$ is unbounded binding.*

4 DVNIZK Proof for NP from DVPRG

4.1 The Hidden Bit Model, and HB Proofs

The hidden bit model is an ideal formalization of a scenario in which both the prover and the verifier have access to a long string of hidden random bits (let us denote with hb the random bits and $t = t(\lambda)$ the length of the hidden string). In this idealized model, the prover can send to the verifier a subset $S \subset [t]$ of the positions of the hidden bits (together with additional informations). The verifier is restricted to inspecting only the bits of hb residing in the locations specified by the prover, while the prover can see hb entirely.

Definition 15. *A non-interactive proof system* HB *in the hidden bit model is a pair of PPT algorithms* $(\mathsf{HB.Prove}, \mathsf{HB.Verify})$ *such that*

- $\mathsf{HB.Prove}(\mathsf{hb}, x, w)$*, on input a random bit string* $\mathsf{hb} \in \{0,1\}^t$*, and a word* $x \in \mathscr{L}$ *with witness* w*, outputs a subset* $S \subset [t]$ *together with a string* M *of auxiliary informations,*
- $\mathsf{HB.Verify}(x, \mathsf{hb}[S], M)$*, on input a word* x*, the subsequence of* hb *indexed by* S*, and an auxiliary information* M*, outputs* $b \in \{0,1\}$*,*

which satisfies the following perfect completeness, ε*-soundness, and (adaptive, single-theorem) zero-knowledge properties:*

- **Perfect Completeness.** *For any* $x \in \mathscr{L}$ *with witness* w*, any* $\mathsf{hb} \in \{0,1\}^t$*, and for* $(S, M) \xleftarrow{\$} \mathsf{HB.Prove}(\mathsf{hb}, x, w)$*, it holds that* $\mathsf{HB.Verify}(x, \mathsf{hb}[S], M) = 1$*.*
- ε**-Soundness.** *For any (possibly unbounded) adversary* $\mathcal{A}$*,*

$$\Pr\left[\begin{array}{l}\mathsf{hb} \xleftarrow{\$} \{0,1\}^t, \\ (x, S, M) \xleftarrow{\$} \mathcal{A}(\mathsf{hb}) \ : \mathsf{HB.Verify}(x, \mathsf{hb}[S], M) = 1 \wedge x \notin \mathscr{L}\end{array}\right] \leq \varepsilon.$$

- **Single-Theorem Zero-Knowledge.** *For any (possibly unbounded) stateful adversary* $\mathcal{A}$*, there exists a simulator* $(\mathsf{Sim}_{zk}, \mathsf{Sim}'_{zk})$ *such that for every* $x \in \mathscr{L}$ *and any* w *satisfying* $R_{\mathscr{L}}(x, w) = 1$*,*
  - *the distributions*

$$\{(\mathsf{hb}[S], S, M) : \mathsf{hb} \xleftarrow{\$} \{0,1\}^t, (S, M) \xleftarrow{\$} \mathsf{HB.Prove}(\mathsf{hb}, x, w)\}$$

    and $\{\mathsf{Sim}_{zk}(x)\}$ *are perfectly indistinguishable;*
  - *the distributions*

$$\{(\mathsf{hb}, S, M) : \mathsf{hb} \xleftarrow{\$} \{0,1\}^t, (S, M) \xleftarrow{\$} \mathsf{HB.Prove}(\mathsf{hb}, x, w)\}$$

    and

$$\{(\mathsf{hb}, S, M) : (\mathsf{hb}[S], S, M) \xleftarrow{\$} \mathsf{Sim}_{zk}(x), \mathsf{hb} \xleftarrow{\$} \mathsf{Sim}'_{zk}(\mathsf{hb}[S], S, M, x, w)\}$$

    are perfectly indistinguishable. That is, the simulator can generate $(\mathsf{hb}[S], S, M)$ *without a witness, and find a* completion *of the hidden string* hb *given a witness* w*, which is identically distributed to an honestly generated hidden string and proof with* w*.*

Note that in the hidden bit model, the parties do not have access to a common random string, but to a string of bits which are perfectly hidden to the verifier until the prover opens a subsequence of them. Therefore, the adaptive and non-adaptive formulations of zero-knowledge are equivalent since the verifier does not get to see anything about hb before producing a word x with a witness w (put differently, it is equivalent to define zero-knowledge for all $x \in \mathscr{L}$ or with respect to adversarially chosen x). Examples of non-interactive proof systems in the hidden-bit model can be found in [17, 28]. We stress that the security of these proof systems is unconditional (although a specific implementation of the HB model can involve cryptography).

4.2 A DVNIZK for NP from any DVPRG

We describe on Figure 2 a general transformation that converts any (unconditional) proof system in the HB model into a DVNIZK for the same language, given any DVPRG. The DVNIZK inherits the specificities of the DVPRG: it satisfies unbounded soundness and/or statistical soundness whenever the DVPRG is unbounded binding and/or statistically binding. At the exception of using a DVPRG instead of a VPRG, the proof system is identical to the one of [16, Section 5.1] (actually, [16] provides a ZAP in the plain model where the first flow can be fixed non-uniformly, which immediately implies a NIZK in the CRS model. Our construction does not imply a ZAP in the plain model, as we need to setup a CRS containing, in particular, the public parameters of the DVPRG. These public parameters must be honestly sampled to maintain the hiding property, hence they cannot be picked by the verifier in the first round.)

While the scheme is almost identical to the scheme of [16], the proof of soundness is more involved, as it must cope with the weaker binding property of our PRGs. To prove soundness, we proceed as follows: we identify a "bad event", which occurs whenever the adversary outputs pvk and a proof π for some position i of correct opening to $1 - x_i$, where $x = \mathsf{Ext}(\mathsf{pvk})$ (Ext being the possibly inefficient extractor guaranteed by the unbounded binding security notion of the DVPRG). We show that when this bad even does *not* happen, then there is a string (essentially $x \oplus \rho$, where ρ is a long random string which is part of the CRS) which is a *bad string*, in the sense that if this string is used as the hidden bit string of the HB proof system, there exists accepting proofs of incorrect statement with respect to this hidden string. Then, we rely on the statistical soundness of the HB proof system to argue that only a tiny fraction of all possible strings (of a given length) are bad strings. Since ρ is random and x is uniquely defined given pvk, we can rely on the fact that pvk is short to argue, with a counting argument, that there is a negligible probability (over the random choice of ρ) that there exists a short pvk such that $\rho \oplus \mathsf{Ext}(\mathsf{pvk})$ is a bad string. Hence, this situation is statistically unlikely, and we must be in the case where the bad event happens; then, we conclude the proof by observing that an occurrence of this bad event directly contradicts the unbounded binding property of the DVPRG. In contrast, the argument of [16] uses a counting argument over all possible *seeds* of the VPRG, which crucially relies on their stronger binding

property which states that any possible pvk is in the stretch of the PRG, and is bound to a seed (while this seed need not be unique, all seeds associated to a given pvk must lead to the same pseudorandom string).

DVNIZK Proof System Π

Let $\mathscr{L}$ be a language and let $y \in \mathscr{L}$ be a word with witness w. Let $\lambda \leftarrow |y|$. The DVNIZK relies on an HB proof system HB for the statement $y \in \mathscr{L}$ which uses $\ell = \ell(\lambda)$ hidden bits, and achieves $2^{-\lambda}$-statistical soundness. Let $\mathcal{G} = (\mathcal{G}.\mathsf{Setup}, \mathcal{G}.\mathsf{Stretch}, \mathcal{G}.\mathsf{Prove}, \mathcal{G}.\mathsf{Verify})$ be a DVPRG, with public verification key size $s(\lambda)$ and output size $m(\lambda)$, satisfying $m > (1 + s/\lambda)\ell + \ell^2/\lambda$. In the following, we consider the HB proof system HB' obtain by executing HB m/ℓ times in parallel (with independent hidden bits) and accepting only if all executions are accepted. Note that HB' uses m hidden bits and achieves $2^{-\lambda m/\ell}$-statistical soundness.

- $\Pi.\mathsf{Setup}(1^\lambda)$: on input the security parameter in unary, compute $(\mathsf{pp}, \mathcal{T}) \xleftarrow{\$} \mathcal{G}.\mathsf{Setup}(1^\lambda, m(\lambda))$, and $\rho \xleftarrow{\$} \{0,1\}^m$. Output $\mathsf{crs} \leftarrow (\mathsf{pp}, \rho)$ and $\mathcal{T}$.
- $\Pi.\mathsf{Prove}(\mathsf{crs}, y, w)$: parse crs as (pp, ρ). Compute $(\mathsf{pvk}, x, \mathsf{aux}) \xleftarrow{\$} \mathcal{G}.\mathsf{Stretch}(\mathsf{pp})$. Pick $\theta \xleftarrow{\$} \{0,1\}^\ell$. For $i = 1$ to m, set $\mathsf{hb}_i \leftarrow x_i \oplus \rho_i \oplus \theta_{(i-1 \bmod \ell)+1}$. Define $\mathsf{hb} = (\mathsf{hb}_i)_i$ to be the hidden string of HB'. Compute an HB proof $(S, M) \xleftarrow{\$} \mathsf{HB}'.\mathsf{Prove}(\mathsf{hb}, y, w)$. For every $i \in S$, compute $\pi_i \xleftarrow{\$} \mathcal{G}.\mathsf{Prove}(\mathsf{pp}, \mathsf{aux}, i)$. Output $(\mathsf{pvk}, \theta, S, \mathsf{hb}[S], M, (\pi_i)_{i \in S})$.
- $\Pi.\mathsf{Verify}(\mathsf{crs}, G, \boldsymbol{\pi}, \mathcal{T})$: parse crs as (pp, ρ) and $\boldsymbol{\pi}$ as $(\mathsf{pvk}, \theta, S, \mathsf{hb}[S], M, (\pi_i)_{i \in S})$. For every $i \in S$, set $x_i \leftarrow \mathsf{hb}_i \oplus \rho_i \oplus \theta_{(i-1 \bmod \ell)+1}$ and check that $\mathcal{G}.\mathsf{Verify}(\mathsf{pp}, \mathsf{pvk}, \mathcal{T}, i, x_i, \pi_i)$. Check that $\mathsf{HB}'.\mathsf{Verify}(y, \mathsf{hb}[S], M)$ returns 1. Output 1 if all checks succeeded, and 0 otherwise.

Fig. 2. Designated-verifier non-interactive zero-knowledge proof system Π for a language $\mathscr{L}$ using a DVPRG $\mathcal{G}$ and an HB proof system HB

Theorem 16. *Let $\mathcal{G}$ be a hiding unbounded binding* DVPRG*, and let $(\Pi.\mathsf{Setup}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Verify})$ be the* DVNIZK *proof system given on Figure 2. Then Π satisfies computational witness-indistinguishability and unbounded adaptive soundness. Furthermore, if $\mathcal{G}$ is equivocable, Π satisfies (adaptive, single-theorem) zero-knowledge.*

The completeness of Π follows immediately from the completeness of HB and $\mathcal{G}$. In the remainder of this section, we prove Theorem 16. The proof of witness-indistinguishability is similar to the one given in [16], but the proof of soundness is more involved (see the previous discussion).

4.3 Witness Indistinguishability of Π

We prove the witness-indistinguishability of Π through a sequence of hybrids. Let $\mathcal{A}$ be a PPT adversary; assume toward contradiction that

$$\left| \Pr\left[\begin{array}{ll}(\mathsf{crs},\mathcal{T}) \xleftarrow{\$} \Pi.\mathsf{Setup}(1^\lambda), & \mathcal{A}(\mathsf{crs},\boldsymbol{\pi})=1\\ (y,w_0,w_1)\xleftarrow{\$}\mathcal{A}(\mathsf{crs}), & :\wedge R_{\mathscr{L}}(y,w_0)=1\\ \boldsymbol{\pi}\xleftarrow{\$}\Pi.\mathsf{Prove}(\mathsf{crs},y,w_0) & \wedge R_{\mathscr{L}}(y,w_1)=1\end{array}\right] - \Pr\left[\begin{array}{ll}(\mathsf{crs},\mathcal{T}) \xleftarrow{\$} \Pi.\mathsf{Setup}(1^\lambda), & \mathcal{A}(\mathsf{crs},\boldsymbol{\pi})=1\\ (y,w_0,w_1)\xleftarrow{\$}\mathcal{A}(\mathsf{crs}), & :\wedge R_{\mathscr{L}}(y,w_0)=1\\ \boldsymbol{\pi}\xleftarrow{\$}\Pi.\mathsf{Prove}(\mathsf{crs},y,w_1) & \wedge R_{\mathscr{L}}(y,w_1)=1\end{array}\right]\right| \geq \varepsilon$$

for some non-negligible quantity ε. Let us denote H_b for $b \in \{0,1\}$ the experiment in which we set $(\mathsf{crs},\mathcal{T}) \xleftarrow{\$} \Pi.\mathsf{Setup}(1^\lambda)$, $(y,w_0,w_1) \xleftarrow{\$} \mathcal{A}(\mathsf{crs})$, $\boldsymbol{\pi} \xleftarrow{\$} \Pi.\mathsf{Prove}(\mathsf{crs},y,w_b)$, and output $b' \xleftarrow{\$} \mathcal{A}(\mathsf{crs},\boldsymbol{\pi})$.

Recall that HB' consists of m/ℓ parallel repetitions of HB (with independent hidden bits hb^j). We consider a sequence of intermediate hybrids $H_{0.j}$ for $j=0$ to m/ℓ, in which we use the witness w_1 for the j first repetitions (computing (S_j,M_j) as $\mathsf{HB}.\mathsf{Prove}(\mathsf{hb}^j,y,w_0)$) and the witness w_0 for the repetitions $j+1$ to m/ℓ. By a standard pigeonhole argument, there exists a j such that the advantage of $\mathcal{A}$ in distinguishing $H_{0.j}$ from $H_{0.j+1}$ is at least $\varepsilon\ell/m$. We further divide $H_{0.j}$ in the following sub-hybrids:

- $H_{0.j.0}$. In this hybrid, we modify the generation of $(\mathsf{hb}^{j+1},S_{j+1},M_{j+1})$. Namely, we compute $(\mathsf{pvk},x,\mathsf{aux}) \xleftarrow{\$} \mathcal{G}.\mathsf{Stretch}(\mathsf{pp})$ (let x^{j+1},ρ^{j+1} denote the $(j+1)$-th block of ℓ bits of x,ρ), generate $(\mathsf{hb}^{j+1}[S_{j+1}],S_{j+1},M_{j+1}) \xleftarrow{\$} \mathsf{Sim}_{\mathsf{zk}}(y)$, $\mathsf{hb}^{j+1} \xleftarrow{\$} \mathsf{Sim}'_{\mathsf{zk}}(\mathsf{hb}^{j+1}[S_{j+1}],S_{j+1},M_{j+1},y,w_1)$, and set $\theta \leftarrow x^{j+1} \oplus \rho_{j+1} \oplus \mathsf{hb}^{j+1}$. The other repetitions of HB are executed as before; note that it holds that $\mathsf{hb}_i = x_i \oplus \rho_i \oplus \theta_{(i-1) \bmod \ell}$ for every $i \leq m$. By the (perfect) single-theorem zero-knowledge property of HB, the distribution of $(\mathsf{crs},\boldsymbol{\pi})$ in $H_{0.j.1}$ is identical to its distribution in $H_{0.j}$, hence the advantage of $\mathcal{A}$ in distinguishing $H_{0.j.1}$ from $H_{0.j+1}$ is at least $\varepsilon\ell/m$.
- $H_{0.j.k}$. We denote by $\ell - r$ the size of S_{j+1} (r is the size of the "unopened" subsequence of hb^{j+1}). For $k=0$ to r, we modify the generation of hb^{j+1} as follows: we generate as before $(\mathsf{hb}^{j+1}[S_{j+1}],S_{j+1},M_{j+1}) \xleftarrow{\$} \mathsf{Sim}_{\mathsf{zk}}(y)$, denote R_{j+1} the set $[\ell] \setminus S_{j+1}$ of unopened positions of hb^{j+1}, and compute
  - $\mathsf{hb}^{j+1.0}[R_{j+1}] \xleftarrow{\$} \mathsf{Sim}'_{\mathsf{zk}}(\mathsf{hb}^{j+1}[S_{j+1}],S_{j+1},M_{j+1},y,w_0)$,
  - $\mathsf{hb}^{j+1.1}[R_{j+1}] \xleftarrow{\$} \mathsf{Sim}'_{\mathsf{zk}}(\mathsf{hb}^{j+1}[S_{j+1}],S_{j+1},M_{j+1},y,w_1)$.

  Then, we define $\mathsf{hb}^{j+1}[R_{j+1}]$ to be the string that agrees with $\mathsf{hb}^{j+1.1}[R_{j+1}]$ for positions 1 to k, and with $\mathsf{hb}^{j+1.0}[R_{j+1}]$ for positions $k+1$ to r. By a standard pingeonhole argument, there exists a k such that $\mathcal{A}$ distinguishes $H_{0.j.k}$ from $H_{0.j.k+1}$ with probability at least $\varepsilon\ell/(mr) \geq \varepsilon/m$.

Note that the string hb^{j+1} differs by a single bit between $H_{0.j.k}$ and $H_{0.j.k+1}$. From there, we immediately reach a contradiction to the hiding property of

$\mathcal{G}$: denoting $i \leftarrow (j+1)\ell + k + 1$, we receive $(\mathsf{pp}, \mathsf{pvk}, i, (x_t, \pi_t)_{t \neq i}$, compute $(\mathsf{hb}^{j+1}[S_{j+1}], S_{j+1}, M_{j+1}) \xleftarrow{\$} \mathsf{Sim}_{\mathsf{zk}}(y)$, guess the value x_i at random (completing the string x), and set $\theta \leftarrow x^{j+1} \oplus \rho_{j+1} \oplus \mathsf{hb}^{j+1}$. Depending on our guess of x_i, the distribution of $(\mathsf{crs}, \boldsymbol{\pi})$ is either identical to its distribution in $H_{0.j.k}$ or in $H_{0.j.k+1}$, hence we distinguish between $x_i = 0$ and $x_i = 1$ with probability at least ε/m. This concludes the proof.

4.4 Adaptive Single-Theorem Zero-Knowledge of Π

A witness-indistinguishable NIZK proof system for NP implies an adaptive zero-knowledge proof system for NP, by the transformation of [17]. However, if $\mathcal{G}$ is equivocable, there is a more direct construction: we prove that in this case, the DVNIZK Π is adaptive single-theorem zero knowledge (and can be made adaptive multi-theorem zero-knowledge using [17]); the argument is simpler than for witness indistinguishability, does only use $\mathsf{Sim}_{\mathsf{zk}}$ (the simulator $\mathsf{Sim}'_{\mathsf{zk}}$ is not needed), and does not require θ (which can be removed from the construction – we keep it in the proof below for simplicity). Let $\mathcal{A}$ be a PPT adversary against the (adaptive) single-theorem zero-knowledge of Π. Let $\mathsf{Sim} = (\mathsf{Sim}_0, \mathsf{Sim}_1)$ be the following simulator:

- On input 1^λ, Sim_0 computes $(\mathsf{pp}, \mathcal{T}) \xleftarrow{\$} \mathcal{G}.\mathsf{SimSetup}(1^\lambda, m)$, and $\rho \xleftarrow{\$} \{0,1\}^m$. He outputs $\mathsf{crs} \leftarrow (\mathsf{pp}, \rho)$ and $\mathcal{T}$.
- On input $(\mathsf{crs}, \mathcal{T}, y)$, Sim parses crs as (pp, ρ) and computes $(\mathsf{pvk}, x', \mathsf{aux}) \xleftarrow{\$} \mathcal{G}.\mathsf{Stretch}(\mathsf{pp})$. Then, Sim_1 runs $(\mathsf{hb}[S], S, M) \xleftarrow{\$} \mathsf{Sim}_{\mathsf{zk}}(y)$, where $\mathsf{Sim}_{\mathsf{zk}}$ is the simulator of the zero-knowledge property of HB'. Sim_1 picks $\theta \xleftarrow{\$} \{0,1\}^\ell$. For every $i \in S$, he sets $x_i \leftarrow \mathsf{hb}_i \oplus \rho_i \oplus \theta_{(i-1 \bmod \ell)+1}$ and computes $\pi_i \xleftarrow{\$} \mathcal{G}.\mathsf{Equivocate}(\mathsf{pp}, \mathsf{pvk}, i, x_i, \mathcal{T})$. Sim_1 outputs $(\mathsf{pvk}, \theta, S, \mathsf{hb}[S], M, (\pi_i)_{i \in S})$.

We prove that

$$\left| \Pr\left[\begin{array}{l} (\mathsf{crs}, \mathcal{T}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda), \\ (y, w) \xleftarrow{\$} \mathcal{A}(\mathsf{crs}, \mathcal{T}), \\ \pi \xleftarrow{\$} \Pi.\mathsf{Prove}(\mathsf{crs}, y, w) \end{array} : (R_{\mathscr{L}}(y, w) = 1) \wedge (\mathcal{A}(\pi) = 1) \right] - \Pr\left[\begin{array}{l} (\mathsf{crs}, \mathcal{T}) \xleftarrow{\$} \mathsf{Sim}_0(1^\lambda), \\ (y, w) \xleftarrow{\$} \mathcal{A}(\mathsf{crs}, \mathcal{T}), \\ \pi \xleftarrow{\$} \mathsf{Sim}_1(\mathsf{crs}, \mathcal{T}, y) \end{array} : (R_{\mathscr{L}}(y, w) = 1) \wedge (\mathcal{A}(\pi) = 1) \right] \right| \approx 0,$$

through a sequence of hybrids.

- **Game H_0.** This is the real game, where we generate $(\mathsf{crs}, \mathcal{T}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$, run $(y, w) \xleftarrow{\$} \mathcal{A}(\mathsf{crs}, \mathcal{T})$, $\pi \xleftarrow{\$} \Pi.\mathsf{Prove}(\mathsf{crs}, y, w)$, and $b \xleftarrow{\$} \mathcal{A}(\pi)$.
- **Game H_1.** In this game, we generate instead $(\mathsf{crs}, \mathcal{T})$ as $\mathsf{Sim}_0(1^\lambda)$ (that is, we compute $(\mathsf{pp}, \mathcal{T}) \xleftarrow{\$} \mathcal{G}.\mathsf{SimSetup}(1^\lambda, m)$ and $\rho \xleftarrow{\$} \{0,1\}^m$). Furthermore, we modify $\Pi.\mathsf{Prove}(\mathsf{crs}, y, w)$ as follow: after computing $(\mathsf{pvk}, x', \mathsf{aux}) \xleftarrow{\$} \mathcal{G}.\mathsf{Stretch}(\mathsf{pp})$, we pick $x \xleftarrow{\$} \{0,1\}^m$ and set $\mathsf{hb}_i \leftarrow x_i \oplus \rho_i \oplus \theta_{(i-1 \bmod \ell)+1}$. We

compute the HB proof (S, M) honestly using (y, w) and the hidden string hb. Finally, we compute the π_i as $\mathcal{G}.\mathsf{Equivocate}(\mathsf{pp}, \mathsf{pvk}, \mathcal{T}, i, x_i)$.
By the equivocability of $\mathcal{G}$, the distribution of $(\mathsf{pp}, \mathsf{pvk}, \mathcal{T}, x, (\pi_i)_{i\in S})$ in H_1 is computationally indistinguishable from its distribution in H_0, and the rest of the proof is computed from $(\mathsf{pp}, \mathsf{pvk}, \mathcal{T}, x)$ identically in both games, hence there is a direct reduction from breaking the equivocability of $\mathcal{G}$ to distinguishing H_0 and H_1.

- **Game H_2.** In this game, instead of picking $x \xleftarrow{\$} \{0,1\}^m$ and setting $\mathsf{hb}_i \leftarrow x_i \oplus \rho_i \oplus \theta_{(i-1 \bmod \ell)+1}$, we first pick $\mathsf{hb} \xleftarrow{\$} \{0,1\}$ and set $x_i \leftarrow \mathsf{hb}_i \oplus \rho_i \oplus \theta_{(i-1 \bmod \ell)+1}$ for every $i \leq m$. Note that this is a purely syntactic change, since hb and x are just a uniformly random sharing of $(\rho_i \oplus \theta_{(i-1 \bmod \ell)+1})_{i\leq m}$, hence this game is perfectly indistinguishable from the previous one.
- **Game H_3.** In this game, we play as in Game H_2 except that we compute $(\mathsf{hb}[S], S, M) \xleftarrow{\$} \mathsf{Sim}_{\mathsf{zk}}(y)$ instead (note that the remaining hidden bits of hb are never used). By the single-theorem zero-knowledge property of HB, this game is perfectly indistinguishable from the previous one. Note that Game H_3 does exactly correspond to the simulation with $(\mathsf{Sim}_0, \mathsf{Sim}_1)$. This concludes the proof.

4.5 Unbounded Adaptive Soundness of Π

Let $\mathcal{A}$ be a PPT adversary against the soundness of Π, which is given oracle access to a verification oracle $\mathcal{O}(\mathsf{crs}, \cdot, \cdot, \mathcal{T})$. Let $(\mathsf{crs}, \mathcal{T}) \xleftarrow{\$} \Pi.\mathsf{Setup}(1^\lambda)$, and parse crs as (pp, ρ). Run $(y, \boldsymbol{\pi}) \xleftarrow{\$} \mathcal{A}(\mathsf{crs})$. Let ε denote the probability (over the coins of $\Pi.\mathsf{Setup}$) that $\Pi.\mathsf{Verify}(\mathsf{crs}, y, \boldsymbol{\pi}, \mathcal{T}) = 1$ and $y \notin \mathscr{L}$:

$$\Pr\left[\begin{matrix}(\mathsf{crs}, \mathcal{T}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda), \\ (y, \boldsymbol{\pi}) \xleftarrow{\$} \mathcal{A}^{\mathcal{O}(\mathsf{crs}, \cdot, \cdot, \mathcal{T})}(\mathsf{crs})\end{matrix} : \Pi.\mathsf{Verify}(\mathsf{crs}, y, \boldsymbol{\pi}, \mathcal{T}) = 1 \wedge y \notin \mathscr{L}\right] = \varepsilon.$$

In the following, we assume for the sake of contradiction that ε is non-negligible. We will construct from $\mathcal{A}$ an adversary $\mathcal{B}$ which contradicts the unbounded binding of $\mathcal{G}$. $\mathcal{B}$ interacts with $\mathcal{A}$ in the unbounded soundness security experiment of Π. The challenger of the unbounded binding property of $\mathcal{G}$ samples $(\mathsf{pp}, \mathcal{T}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, m)$. $\mathcal{B}$ receives pp and is given oracle access to $\mathcal{G}.\mathsf{Verify}(\mathsf{pp}, \cdot, \mathcal{T}, \cdot, \cdot, \cdot)$. It picks $\rho \xleftarrow{\$} \{0,1\}^m$, sets $\mathsf{crs} \leftarrow (\mathsf{pp}, \rho)$, and runs $\mathcal{A}(\mathsf{crs})$. Let q be the number of queries that $\mathcal{A}$ asks to $\mathcal{O}(\mathsf{crs}, \cdot, \cdot, \mathcal{T})$ in the unbounded soundness security experiment of Π. $\mathcal{B}$ simulates the answers of $\mathcal{O}(\mathsf{crs}, \cdot, \cdot, \mathcal{T})$ as follows: on input $\boldsymbol{\pi} = (\mathsf{pvk}, \theta, S, \mathsf{hb}[S], M, (\pi_i)_{i\in S})$ it sets for every $i \in S$ $x_i \leftarrow \mathsf{hb}_i \oplus \rho_i \oplus \theta_{(i-1 \bmod \ell)+1}$ and calls $\mathcal{G}.\mathsf{Verify}(\mathsf{pp}, \cdot, \mathcal{T}, \cdot, \cdot, \cdot)$ on input $(\mathsf{pvk}, i, x_i, \pi_i)$. Then, it verifies the HB proof $(S, \mathsf{hb}[S], M)$ for the statement $y \in \mathscr{L}$ and outputs 1 iff all checks succeeded. Then, $\mathcal{A}$ outputs a pair $(y, \boldsymbol{\pi})$. Since $\mathcal{B}$ perfectly simulates crs and the answers of $\mathcal{O}(\mathsf{crs}, \cdot, \cdot, \mathcal{T})$, it holds that $\Pi.\mathsf{Verify}(\mathsf{crs}, y, \boldsymbol{\pi}, \mathcal{T}) = 1 \wedge y \notin \mathscr{L}$ with probability ε over the coins of the challenger and $\mathcal{A}, \mathcal{B}$. Finally, $\mathcal{B}$ parses $\boldsymbol{\pi}$ as $(\mathsf{pvk}^*, \theta, S, \mathsf{hb}[S], M, (\pi_i)_{i\in S})$, picks $i^* \xleftarrow{\$} S$, and outputs $(\mathsf{pvk}^*, i^*, \pi_{i^*})$. To simplify the analysis in the following, we assume that $\mathcal{B}$ also outputs $(\mathsf{crs}, y, \boldsymbol{\pi})$ in addition to $(\mathsf{pvk}^*, i^*, \pi_{i^*})$ (it is only a

syntactic modification that will make it more convenient to describe the probability experiments).

We analyze the probability that $\mathcal{G}.\mathsf{Verify}(\mathsf{pp}, \mathsf{pvk}^*, \mathcal{T}, i^*, 1 - x_{i^*}, \pi_{i^*}) = 1$. Let us call 'bad' a string hb for which there exists $y \notin \mathscr{L}$ and an accepting proof of $y \in \mathscr{L}$ under the HB proof system HB'. Under the $2^{-\lambda m/\ell}$-statistical soundness of HB', the ratio of bad strings must be at most $2^{-\lambda m/\ell} < 2^{\lambda+s+\ell}$. Let us say that a string hb' is "close to a bad string w.r.t. pp" hb if there exists $\theta \in \{0,1\}^\ell$ and a public verification key $\mathsf{pvk} \in \{0,1\}^{s(\lambda)}$ such that $\mathsf{hb}' = (x_i \oplus \mathsf{hb}_i \oplus \theta_{(i-1 \bmod \ell)+1})_i$ is a bad string, where $x = \mathsf{Ext}(\mathsf{pp}, \mathsf{pvk})$. As there are at most $2^{\ell+s}$ possible choices of (θ, pvk), for any choice of public parameters pp, the ratio of strings which are close to a bad string w.r.t. pp must be at most $2^{-\lambda-s-\ell} \cdot 2^{\ell+s} = 2^{-\lambda}$. Therefore, with overwhelming probability $1 - 2^{-\lambda}$ over the distribution of ρ, ρ is not close to a bad string w.r.t. pp, hence there does not exist a string $(y, \mathsf{pvk}, \theta)$ with $y \notin \mathscr{L}$ such that $(\mathsf{hb}_i)_i = (x_i \oplus \rho_i \oplus \theta_{(i-1 \bmod \ell)+1})_i$ is a bad string.

We consider two complementary cases, one of which must necessarily occur:

Case 1. With probability at least $\varepsilon/2$, the output $(y, \boldsymbol{\pi})$ of $\mathcal{A}$ satisfies $\mathit{\Pi}.\mathsf{Verify}(\mathsf{crs}, y, \boldsymbol{\pi}, \mathcal{T}) = 1 \wedge y \notin \mathscr{L}$, and for every $i \in S$, it holds that $\mathcal{G}.\mathsf{Verify}(\mathsf{pp}, \mathsf{pvk}^*, \mathcal{T}, i^*, 1 - x_i, \pi_i) = 1$. That is,

$$\Pr\left[\begin{array}{ll} (\mathsf{pp}, \mathcal{T}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, m), & \mathit{\Pi}.\mathsf{Verify}(\mathsf{crs}, y, \boldsymbol{\pi}, \mathcal{T}) = 1 \\ (\mathsf{pvk}^*, i^*, \pi_{i^*}, \mathsf{crs}, y, \boldsymbol{\pi}) \xleftarrow{\$} \mathcal{B}(\mathsf{pp}), & : \wedge\, y \notin \mathscr{L} \wedge \forall i \in S, \\ x \leftarrow \mathsf{Ext}(\mathsf{pp}, \mathsf{pvk}^*) & \mathcal{G}.\mathsf{Verify}(\mathsf{pp}, \mathsf{pvk}^*, \mathcal{T}, i, 1 - x_i, \pi_i) = 1 \end{array}\right] \geq \frac{\varepsilon}{2},$$

where $\mathcal{B}$ is given oracle access to $\mathcal{G}.\mathsf{Verify}(\mathsf{pp}, \cdot, \mathcal{T}, \cdot, \cdot, \cdot)$. Now, parse $\boldsymbol{\pi}$ as $(\mathsf{pvk}^*, \theta, S, \mathsf{hb}[S], M, (\pi_i)_{i \in S})$. Let x' denote $\mathsf{Ext}(\mathsf{pp}, \mathsf{pvk}^*)$, and let $(\mathsf{hb}'_i)_i = (x'_i \oplus \rho_i \oplus \theta_{(i-1 \bmod \ell)+1})_i$. Since a random ρ has probability at most $1/2^\lambda$ to be close to a bad string w.r.t. pp, $(\mathsf{hb}'_i)_i$ has probability at most $1/2^\lambda$ of being a bad string. Therefore, if case 1 happens, we necessarily have (denoting $\mu = \varepsilon/2 - 1/2^\lambda$):

$$\Pr\left[\begin{array}{ll} (\mathsf{pp}, \mathcal{T}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, m), & \mathit{\Pi}.\mathsf{Verify}(\mathsf{crs}, y, \boldsymbol{\pi}, \mathcal{T}) = 1 \\ (\mathsf{pvk}^*, i^*, \pi_{i^*}, \mathsf{crs}, y, \boldsymbol{\pi}) \xleftarrow{\$} \mathcal{B}(\mathsf{pp}), & \wedge\, y \notin \mathscr{L} \wedge \forall i \in S, \\ x \leftarrow \mathsf{Ext}(\mathsf{pp}, \mathsf{pvk}^*) & : \mathcal{G}.\mathsf{Verify}(\mathsf{pp}, \mathsf{pvk}^*, \mathcal{T}, i, 1 - x_i, \pi_i) = 1 \\ & \wedge\, \mathsf{hb}' \text{ is not a bad string} \end{array}\right] \geq \mu.$$

Note that the condition $\mathit{\Pi}.\mathsf{Verify}(\mathsf{crs}, y, \boldsymbol{\pi}, \mathcal{T}) = 1$ implies that for all $i \in S$, denoting $x_i \leftarrow \mathsf{hb}_i \oplus \rho_i \oplus \theta_{(i-1 \bmod \ell)+1}$, $\mathcal{G}.\mathsf{Verify}(\mathsf{pp}, \mathsf{pvk}^*, \mathcal{T}, i, x_i, \pi_i) = 1$. Denoting $x' = \mathsf{Ext}(\mathsf{pp}, \mathsf{pvk}^*)$, it holds by assumption that $\mathcal{G}.\mathsf{Verify}(\mathsf{pp}, \mathsf{pvk}^*, \mathcal{T}, i, 1 - x_i, \pi_i) = 1$ for every $i \in S$, hence $\mathcal{G}.\mathsf{Verify}(\mathsf{pp}, \mathsf{pvk}^*, \mathcal{T}, i, 1 - x'_i, \pi_i) = 0$. This implies that for any $i \in S$, $x_i \neq 1 - x'_i$, hence that $(x_i)_{i \in S} = (x'_i)_{i \in S}$, which in turns implies that $\mathsf{hb}[S] = \mathsf{hb}'[S]$. Therefore, if case 1 happens, with probability at least $\varepsilon/2 - 1/2^\lambda$ we have the following:

- $y \notin \mathscr{L}$,
- $\mathit{\Pi}.\mathsf{Verify}(\mathsf{crs}, y, \boldsymbol{\pi}, \mathcal{T}) = 1$,
- $\mathsf{hb}[S] = \mathsf{hb}'[S]$ is not a bad string.

However, by the soundness of the HB proof system HB', there cannot exist any accepting proof $(S, \mathsf{hb}[S], M)$ for a statement $y \notin \mathscr{L}$ unless $\mathsf{hb}[S]$ is a bad string. Since $\Pi.\mathsf{Verify}$ does also check the HB proof, this event can never happen and we get:

$$\frac{\varepsilon}{2} - \frac{1}{2^\lambda} = 0,$$

contradicting our assumption that ε is non-negligible. Hence, case 1 never happens and the following case necessarily happens:

Case 2. There exists $i \in S$ such that $\mathcal{G}.\mathsf{Verify}(\mathsf{pp}, \mathsf{pvk}^*, \mathcal{T}, i, 1 - x_i, \pi_i) = 1$ with probability at least $\varepsilon/2$. That is,

$$\Pr\left[\begin{matrix}(\mathsf{pp}, \mathcal{T}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, m), \\ (\mathsf{pvk}^*, i^*, \pi_{i^*}, \mathsf{crs}, y, \boldsymbol{\pi}) \xleftarrow{\$} \mathcal{B}(\mathsf{pp})\end{matrix} : \begin{matrix}\Pi.\mathsf{Verify}(\mathsf{crs}, y, \boldsymbol{\pi}, \mathcal{T}) = 1 \\ \wedge\, y \notin \mathscr{L} \wedge \exists i \in S, \\ \mathcal{G}.\mathsf{Verify}(\mathsf{pp}, \mathsf{pvk}^*, \mathcal{T}, i, 1 - x_i, \pi_i) = 1\end{matrix}\right] \geq \frac{\varepsilon}{2}.$$

Since $\mathcal{B}$ picks i^* at random in a set S of size at most m, this gives us in particular

$$\Pr\left[\begin{matrix}(\mathsf{pp}, \mathcal{T}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, m), \\ (\mathsf{pvk}^*, i^*, \pi_{i^*}, \mathsf{crs}, y, \boldsymbol{\pi}) \xleftarrow{\$} \mathcal{B}(\mathsf{pp})\end{matrix} : \mathcal{G}.\mathsf{Verify}(\mathsf{pp}, \mathsf{pvk}^*, \mathcal{T}, i^*, 1 - x_i, \pi^*) = 1\right] \geq \frac{\varepsilon}{2m},$$

which immediately gives a contradiction to the unbounded binding of $\mathcal{G}$, concluding the proof.

Impact on our LWE-Based Instantiation. Note that our alternative proof strategy, which does not use any assumed structure for pvk except a bound on its length, is the key to our LWE-based instantiation. Indeed, if we had to assume some structure of pvk (such as "pvk was honestly generated"), we would have to include a NIZK proof of validity of pvk in our instantiation (which is similar to the NIZK proof of validity for the public key used in [37]). Since there might not exist more than a single witness for the validity of pvk, it seems unlikely that we could use a NIWI instead of a NIZK here. By removing entirely the need for proving validity of pvk in our LWE-based instantiation, we enable the construction of a NIZK for NP from LWE using only a NIWI for a simple language (bounded distance decoding), improving over the result of [37].

5 Constructions of Designated-Verifier Pseudorandom Generators

5.1 A DVPRG from the CDH Assumption

Assumptions. Let DHGen denote a PPT algorithm which, on input 1^λ, outputs an integer n, the description of a group $\mathbb{G}$ of order n, and a generator g of $\mathbb{G}$. The *computational Diffie-Hellman assumption* (CDH), with respect to g over $\mathbb{G}$, states that it is computationally infeasible, for any PPT algorithm which

is given $(n, g, \mathbb{G})$ and a random pair (g^a, g^b) from $\mathbb{G}^2$, to compute g^{ab}. The *decisional Diffie-Hellman assumption* (DDH), with respect to g over $\mathbb{G}$, states that it is computationally infeasible for any PPT algorithm to distinguish the distribution $\{(g^a, g^b, g^{ab}) \mid (a, b) \xleftarrow{\$} \mathbb{Z}_n^3\}$ of random DDH tuples from the uniform distribution over $\mathbb{G}^3$.

The *twin (computational or decisional) Diffie-Hellman assumption* (twin-CDH and twin-DDH), defined in [10], are variants of the CDH and DDH assumptions. The twin-CDH problem with respect to g over $\mathbb{G}$ states that it is computationally infeasible, for any PPT algorithm which is given $(n, g, \mathbb{G})$ and a random triple (g^a, g^b, g^c) from $\mathbb{G}^3$, to compute (g^{ab}, g^{ac}); twin-DDH is its natural decisional variant. Twin-CDH (resp. twin-DDH) is equivalent to the standard CDH (resp. DDH) assumption. However, there is a natural trapdoor test that allows to check the correctness of twin-DDH tuples: let (α, β) be a random pair of exponents satisfying $g^c = g^\alpha (g^b)^{-\beta}$ (note that many such pairs exist). Then given an input (g^a, g^b, g^c), the probability for an arbitrary (possibly unbounded) adversary $\mathcal{A}$ to output a pair (h_1, h_2) such that the truth value of $h_1^\beta h_2 = (g^a)^\alpha$ does not agree with the truth value of $(h_1 = g^{ab}) \wedge (h_2 = g^{ac})$ is at most $1/n$ (see [10]). Therefore, a verifier which is given the trapdoor (α, β) can check the correctness of a twin Diffie-Hellman tuple, with negligible error probability. This trapdoor test implies that the *gap* twin Diffie-Hellman problem, which states that solving the twin-CDH problem is hard even given an oracle that solves the twin-DDH problem, is at least as hard as the standard CDH problem.

Our Construction. Our construction will rely on the conjectured hardness of the computational Diffie-Hellman (CDH) assumption. Let $B : \mathbb{G}^3 \mapsto \{0, 1\}$ be a predicate satisfying the following property: given (g^a, g^b, g^c), computing $B(g^a, g^{ab}, g^{ac})$ should be as hard (up to polynomial factors) as computing (g^a, g^{ab}, g^{ac}). Note that this implies that distinguishing $B(g^a, g^{ab}, g^{ac})$ from a random bit given a random triple (g^a, g^b, g^c) is as hard as solving CDH. There are standard method to build this predicate using e.g. the Goldreich-Levin construction [18], see e.g. [10] for an illustration in the specific case of CDH. Our construction proceeds as follows:

- $\mathsf{Setup}(1^\lambda, m)$: sample $(n, \mathbb{G}, g) \xleftarrow{\$} \mathsf{DHGen}(1^\lambda)$. For $i = 1$ to m, pick $(a_i, b_i) \xleftarrow{\$} \mathbb{Z}_n^2$ and set $(u_i, v_i) \leftarrow (g^{a_i}, g^{b_i})$. Set $\mathsf{pp} = (u_i, v_i)_{i \leq m}$. For $i = 1$ to m, pick $\beta_i \xleftarrow{\$} \mathbb{Z}_n$ and set $\alpha_i \leftarrow b_i + a_i \beta_i$ (observe that (α_i, β_i) are uniformly distributed exponents subject to $v_i = g^{\alpha_i} u_i^{-\beta_i}$). Output pp and $\mathcal{T} \leftarrow (\alpha_i, \beta_i)_{i \leq m}$. We also define $\mathsf{SimSetup}(1^\lambda, m)$ to be identical to $\mathsf{Setup}(1^\lambda, m)$ and define $\mathcal{T}_s = \mathcal{T}$.
- $\mathsf{Stretch}(\mathsf{pp})$: pick $r \xleftarrow{\$} \mathbb{Z}_n$, set $\mathsf{pvk} \leftarrow g^r$, and for $i = 1$ to m, set $x_i \xleftarrow{\$} B(\mathsf{pvk}, u_i^r, v_i^r)$. Output $(\mathsf{pvk}, x, \mathsf{aux} = r)$.
- $\mathsf{Prove}(\mathsf{pp}, \mathsf{aux}, i)$: output $\pi \leftarrow (u_i^r, v_i^r)$.
- $\mathsf{Equivocate}(\mathsf{pp}, \mathsf{pvk}, \mathcal{T}_s, i, \sigma)$: pick $u' \xleftarrow{\$} \mathbb{G}$, set $v' = \mathsf{pvk}^{\alpha_i} (u')^{-\beta_i}$, and check whether $B(\mathsf{pvk}, u', v') = \sigma$; if it does not hold, start again. Output $\pi \leftarrow (u', v')$.

– $\mathsf{Verify}(\mathsf{pp}, \mathsf{pvk}, \mathcal{T}, i, \sigma, \pi)$: parse π as (u', v'), check whether $B(\mathsf{pvk}, u', v') = \sigma$ and check whether $(u')^{\beta_i} v' = \mathsf{pvk}^{\alpha_i}$. If both checks pass, output 1; otherwise, output 0.

This construction follows the twin Diffie-Hellman paradigm of Cash, Kiltz, and Shoup [10], which relies on the fact that computing the *twin Diffie-Hellman function*, which on input (g^x, g^{y_1}, g^{y_2}) outputs (g^{xy_1}, g^{xy_2}), is at least as hard as solving the CDH problem, even given an oracle for twin-DDH.

Theorem 17. *If the CDH assumption holds over $\mathbb{G}$, then the above construction is a computationally hiding unbounded statistically binding* DVPRG*. Furthermore, if the DDH assumption holds over $\mathbb{G}$, the above construction is also equivocable.*

Proof. Completeness follows easily by inspection. We now look at the unbounded binding property; by Theorem 14, it suffices to show that the scheme is binding, consistent, and that we can efficiently sample trapdoors consistent with pp (a proof of Theorem 14 is given in the full version of this paper [12]). From the analysis of [10, Section 2], as (g^x, g^{y_1}, g^{y_2}) uniquely define the pair (h_1, h_2) such that $(h_1 = g^{xy_1}) \wedge (h_2 = g^{xy_2})$ and any adversary has negligible probability $1/n$ of outputting a non-twin-DH pair (h_1, h_2) that fools the test, it follows that the scheme is statistically binding (the inefficient extractor Ext simply extracts r from $\mathsf{pvk} = g^r$ and computes the string x as $x_i \xleftarrow{\$} B(\mathsf{pvk}, u_i^r, v_i^r)$ for $i = 1$ to m). Second, observe that we can efficiently sample trapdoors consistent with pp, by storing the random values $(a_i, b_i)_i$ and sampling each trapdoor $\mathcal{T} = (\alpha_i, \beta_i)$ as $\beta_i \xleftarrow{\$} \mathbb{Z}_n$ and $\alpha_i \leftarrow b_i + a_i\beta_i$. Therefore, this defines an efficiently sampleable distribution $\mathtt{Dist}((a_i, b_i)_i)$.

We now show that our construction satisfies consistency. Let $\varepsilon \leftarrow 2/n^5$ and let $\mathcal{A}$ be an adversary that, on input $\mathsf{pp} = (u_i, v_i)_i = (g^{a_i}, g^{b_i})_i$, outputs a 4-tuple $(\mathsf{pvk}, i, \pi = (u', v'), \sigma)$ such that

$$\Pr[\beta_i \xleftarrow{\$} \mathbb{Z}_n : \mathsf{Verify}(\mathsf{pp}, \mathsf{pvk}, (\alpha_i, b_i + a_i\beta_i), i, \sigma, (u', v')) = 1] \geq \varepsilon.$$

The above implies that $\mathcal{A}$ outputs (pvk, u', v') such that $(u')^{\beta_i} v' = \mathsf{pvk}^{\alpha_i}$ holds with probability at least $2/n$. Suppose now that $(\mathsf{pvk}, u_i, v_i, u', v')$ is not a twin-DH tuple; let us denote $\mathsf{pvk} = g^r$ and $(u', v') = (g^s, g^t)$ with $s \neq a_i r$ or $t \neq b_i r$. Then the previous equation becomes $g^{s\beta_i + t} = g^{r\alpha_i}$, which gives

$$\beta_i(s - a_i r) = r b_i - t.$$

However, if $s - a_i r \neq 0$ or $r b_i - t \neq 0$, then this equation holds with probability at most $1/n$ over the random choice of β_i, hence since we assumed that this equation is satisfied with probability at least $2/n$, it must be that $s - a_i r = r b_i - t = 0$, hence $(\mathsf{pvk}, u_i, v_i, u', v')$ is a twin-DH tuple. But then, it immediately follows that the above equation is *always* satisfied, independently of the choice of β_i:

$$\Pr[\beta_i \xleftarrow{\$} \mathbb{Z}_n : \mathsf{Verify}(\mathsf{pp}, \mathsf{pvk}, (\alpha_i, b_i + a_i\beta_i), i, \sigma, (u', v')) = 1] = 1,$$

[5] Since n is the order of $\mathbb{G}$ and $\mathbb{G}$ is a group in which CDH is assumed to hold, $2/n$ is negligible in the security parameter.

which concludes the proof of consistency. Since the DVPRG is also statistically binding (in a bounded sense), we use Theorem 14 to conclude that the above construction satisfies (statistical) unbounded binding.

We now discuss the hiding property. We show that a PPT adversary against the hiding property of the above scheme implies the existence of a PPT adversary that solves the computational twin Diffie-Hellman problem. The result follows from the proof of [10] that the computational twin Diffie-Hellman problem is at least as hard as the CDH problem. The reduction is relatively straightforward: given a position $i \leq m$, we pick $(a_j, b_j)_{j \neq i}$, receive a computational twin-DH challenge (c_0, c_1, c_2), and set $\mathsf{pvk} \leftarrow c_0$, $(u_j, v_j) \leftarrow (g^{a_j}, g^{b_j})$ for every $j \neq i$, and $(u_i, v_i) \leftarrow (c_1, c_2)$. We output $\mathsf{pp} \leftarrow (u_j, v_j)_{j \leq m}$, pvk, and $(x_j, \pi_j) \xleftarrow{\$} (B(\mathsf{pvk}, c_0^{a_i}, c_0^{b_j}), (c_0^{a_i}, c_0^{b_j}))$ for every $j \neq i$. Note that $\mathsf{pp}, \mathsf{pvk}$ and the x_j, π_j are distributed exactly as in an honest execution of the experiment. Then, we run $\mathcal{A}(\mathsf{pp}, \mathsf{pvk}, (x_j, \pi_j)_{j \neq i})$ and get a bit b. If $\mathcal{A}$ guesses the value of $x_i = B(\mathsf{pvk}, c_1', c_2')$, where $(c_1', c_2') = (c_1^r, c_2^r)$ for the value r such that $c_0 = \mathsf{pvk} = g^r$, then we efficiently find a hardcore bit for the twin-DH problem with non-negligible probability. As guessing a hardcore bit for twin-DH is at least as hard as solving the computational twin-DH problem, the proof follows.

Regarding equivocability, the reduction gets a DDH challenge (c_0, c_1, c_2). It sets $\mathsf{pvk} \leftarrow g^r$, samples $(\alpha_i, \beta_i)_i \xleftarrow{\$} \mathbb{Z}_n^{2m}$ and $\mathsf{pp} = (u_i, v_i)_i$ as $(c_1^{a_i}, g^{\alpha_i} u_i^{-\beta_i})_i$ with random a_i's. It computes each proof π_i as $(u', v') \leftarrow (c_2^{a_i}, \mathsf{pvk}^{\alpha_i}(u')^{-\beta_i})$. Observe that the distribution of $(\mathsf{pp}, \mathcal{T}, \mathsf{pvk}, (\pi_i)_i)$ is identical to the distribution obtained with an honest run of the DVPRG when (c_0, c_1, c_2) is a DDH tuple, and identical to a run of the DVPRG with the algorithm Equivocate when (c_0, c_1, c_2). Hence, distinguishing honest proofs from equivocated proofs is equivalent to breaking the DDH assumption.

Corollary 18. *Assuming the computational Diffie-Hellman assumption, there exists an unbounded designated-verifier non-interactive (adaptive, multi-theorem) zero-knowledge proof system for NP.*

Note that the above construction also implies that the existence of a (publicly verifiable) NIZK proof system for the DDH language (together with the CDH assumption) would imply a NIZK proof system for NP.

5.2 A DVPRG from the LWE Assumption

We also give a construction of a DVPRG in the LWE setting. Our construction already assumes a designated-verifier NIWI proof system Π for the LWE language. We stress, however, that Π does not have to enjoy zero-knowledge; witness-indistinguishability is sufficient. We also note that Π can be publicly verifiable, in which case the DVPRG becomes publicly verifiable.

Algebraic setting. We largely follow the presentation of [22] and abstract the setting as far as possible. In the following, let $n, m = \mathsf{poly}(\lambda)$ and $q, \beta = 2^{\mathsf{poly}(\lambda)}$

with $m > n$ and $q > \beta$ be suitable integers. We also assume an error distribution χ that outputs integers e with $|e| < \beta$.

The *Learning With Errors (LWE)* problem (relative to n, m, q, β) is to distinguish access to an oracle $\mathcal{O}^{\mathsf{lwe}}_{\mathsf{real},\mathbf{s}}$ (with hardwired uniform $\mathbf{s} \in \mathbb{Z}_q^n$) from access to another oracle $\mathcal{O}^{\mathsf{lwe}}_{\mathsf{rand}}$. Here, $\mathcal{O}^{\mathsf{lwe}}_{\mathsf{real},\mathbf{s}}$ (parameterized over $\mathbf{s} \in \mathbb{Z}_q^n$) outputs samples $(\mathbf{a}, \mathbf{s}^\top \mathbf{a} + e)$ for fresh $\mathbf{a} \xleftarrow{\$} \mathbb{Z}_q^n$ and $e \leftarrow \chi$, and $\mathcal{O}^{\mathsf{lwe}}_{\mathsf{real},\mathbf{s}}$ outputs $(\mathbf{a}, r)$ with fresh $\mathbf{a} \xleftarrow{\$} \mathbb{Z}_q^n$ and $r \xleftarrow{\$} \mathbb{Z}_q$. The LWE assumption is that for every PPT adversary $\mathcal{A}$,

$$\left| \Pr\left[\mathcal{A}^{\mathcal{O}^{\mathsf{lwe}}_{\mathsf{real},\mathbf{s}}}(1^\lambda) = 1 \right] - \Pr\left[\mathcal{A}^{\mathcal{O}^{\mathsf{lwe}}_{\mathsf{rand}}}(1^\lambda) = 1 \right] \right| \approx 0,$$

where the probability is over $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$ and the random coins of $\mathcal{A}$ and the oracles.

In the following, let $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, and consider the language

$$\mathcal{L}_{\mathbf{A}} := \left\{ \mathbf{A}\mathbf{u} \mid \mathbf{u} \in \mathbb{Z}_q^m \text{ with } ||\mathbf{u}||_\infty < \beta \right\}.$$

Depending on $\mathbf{A}$, $\mathcal{L}_{\mathbf{A}}$ may be trivial. However, if $\mathbf{A}$ can be written as $\mathbf{A} = \begin{pmatrix} \mathbf{A}' \\ \mathbf{s}^\top \mathbf{A}' + \mathbf{e} \end{pmatrix}$ with $||\mathbf{e}||_\infty < \beta$, then $\mathcal{L}_{\mathbf{A}}$ consists of all zero-encryptions under Regev's encryption scheme [36]. In that case, $\mathcal{L}_{\mathbf{A}}$ is hard to decide under the LWE assumption.

Homomorphic commitments. In the setting above, Gorbunov, Vaikuntanathan, and Wichs [22] construct homomorphic trapdoor functions (HTDFs). As they point out, HTDFs can also be viewed as homomorphic commitments. Formally, an HTDF HF consists of the following PPT algorithms:

Key generation. $\mathsf{HF.Setup}(1^\lambda)$ outputs a keypair $(\mathsf{pk}, \mathsf{sk})$. We require that pk defines input, output, and index sets $\mathcal{U}$, $\mathcal{V}$, and $\mathcal{X}$. These sets must be efficiently decidable, and we assume are efficiently samplable distributions $D_{\mathcal{U}}$ and $D_{\mathcal{V}}$ over $\mathcal{U}$ and $\mathcal{V}$.

Function evaluation. $f_{\mathsf{pk},x}$ evaluates a deterministic function from $\mathcal{U}$ to $\mathcal{V}$. We can view $f_{\mathsf{pk},x}(u)$ as a commitment under pk to x with random coins u.

Function inversion. $\mathsf{Inv}_{\mathsf{sk},x}$ probabilistically samples a preimage of $f_{\mathsf{pk},x}$. We require that for every $(\mathsf{pk}, \mathsf{sk})$ in the range of $\mathsf{HF.Setup}$, every $x \in \mathcal{X}$, and every v in the range of $f_{\mathsf{pk},x}$, the value $\mathsf{Inv}_{\mathsf{sk},x}(v)$ is distributed statistically close to a random preimage of v under $f_{\mathsf{pk},x}$ sampled from $D_{\mathcal{U}}$.

Homomorphic evaluation. Eval^{in} and Eval^{out} allow homomorphic computations on inputs and outputs, in the following sense. For all pk in the range of $\mathsf{HF.Setup}$, all $\ell \in \mathbb{N}$, all functions g (represented as circuits), all $(x_i, u_i, v_i) \in \mathcal{X} \times \mathcal{U} \times \mathcal{V}$ $(1 \le i \le \ell)$ with $v_i = f_{\mathsf{pk},x_i}(u_i)$, and for $u^* := \mathsf{Eval}^{in}_{\mathsf{pk}}(g, (x_i, u_i)_{i=1}^\ell)$ and $v^* := \mathsf{Eval}^{out}_{\mathsf{pk}}(g, (v_i)_{i=1}^\ell)$, we have

$$f_{\mathsf{pk},g(x_1,\ldots,x_\ell)}(u^*) = v^*.$$

Dual-mode homomorphic commitments. For security, [22] require that it is computationally hard to find (x, u, x', u') with $x \neq x'$ and $f_{\mathsf{pk},x}(u) = f_{\mathsf{pk},x'}(u')$. When

viewing HTDFs as commitment schemes, this corresponds to a computational binding property. For our purposes, however, we require a stronger property that [22] mention but do not formally define or use. Namely, in analogy to dual-mode commitment schemes [26], we require that there are two computationally indistinguishable ways to sample public keys: one way leads to a statistically hiding commitment scheme, and the other to a statistically binding scheme. In the HTDF setting, this translates to the following requirements:

Statistically hiding. For any fixed pk in the range of $\mathsf{HF.Setup}$, and any $x, x' \in \mathcal{X}$, the random variables $f_{\mathsf{pk},x}(u)$ and $f_{\mathsf{pk},x'}(u)$ (for random $u \leftarrow D_{\mathcal{U}}$) are statistically close.

Perfectly binding under alternate key generation. There exists a PPT algorithm $\mathsf{HF.Setup}_{\mathsf{bind}}$ that outputs public keys $\mathsf{pk}_{\mathsf{bind}}$ with the following properties:

- $\mathsf{pk}_{\mathsf{bind}} \overset{c}{\approx} \mathsf{pk}$ for public keys pk output by $\mathsf{HF.Setup}$,
- the "function evaluation" and "homomorphic evaluation" properties above also hold (perfectly) for public keys $\mathsf{pk}_{\mathsf{bind}}$,
- for all $\mathsf{pk}_{\mathsf{bind}}$ in the range of $\mathsf{HF.Setup}_{\mathsf{bind}}$, and all $x, x' \in \mathcal{X}$ with $x \neq x'$, the sets $\{f_{\mathsf{pk}_{\mathsf{bind}},x}(u) \mid u \in \mathcal{U}\}$ and $\{f_{\mathsf{pk}_{\mathsf{bind}},x'}(u) \mid u \in \mathcal{U}\}$ are disjoint. In other words, there are no (x, u, x', u') with $x \neq x'$ and $f_{\mathsf{pk}_{\mathsf{bind}},x}(u) = f_{\mathsf{pk}_{\mathsf{bind}},x'}(u')$.

The instantiation of Gorbunov, Vaikuntanathan, and Wichs. [22] offer a *leveled* instantiation of dual-mode homomorphic commitments. That is, their construction only allows for an arbitrary, but a-priori bounded number of homomorphic base operations on commitments. If this number of operations is exceeded, correctness will cease to hold. For our purposes, this leveled construction is sufficient, since the number and type of homomorphic operations is known in advance.

We further note that their HTDF application does not require any dual-mode features. However, in [22, App. B], they explicitly describe and analyze what we call $\mathsf{HF.Setup}_{\mathsf{bind}}$ above. They show that their construction is secure (in the sense above) under the LWE assumption.

We will not need to consider any specifics of their construction, except for one. Namely, in their scheme, $\{0,1\} \subset \mathcal{X} \subset \mathbb{Z}$, and commitments to x are of the form $f_{\mathsf{pk},x}(u) = \mathbf{A}\mathbf{U} + x\mathbf{G}$ for fixed $\mathbf{A}, \mathbf{G} \in \mathbb{Z}_q^{n \times m}$ defined in pk, and a short $\mathbf{U} \in \mathbb{Z}_q^{m \times m}$ with $||\mathbf{U}||_\infty < \beta$. In other words, commitments to $x = 0$ are composed of m elements of the language $\mathcal{L}_{\mathbf{A}}$ defined above. Furthermore, a preimage u is the corresponding witness $\mathbf{U}$. Hence, given an argument system for $\mathcal{L}_{\mathbf{A}}$, we can prove that a given commitment v commits to a given x by proving that $v - x\mathbf{G} \in \mathcal{L}_{\mathbf{A}}^m$.

Our construction. We can now give our construction of a DVPRG. We assume dual-mode homomorphic commitments HF as described above, and any family of PRGs $G_m : \{0,1\}^\lambda \to \{0,1\}^m$. In the following, let $G_{m,i}$ denote the circuit that computes the i-th output bit of G_m. Furthermore, we assume a NIWI proof

system $\Pi = (\Pi.\mathsf{Gen}, \Pi.\mathsf{Prove}, \Pi.\mathsf{Verify})$ for the language $\mathcal{L}_{\mathbf{A}}$. Slightly abusing notation, we will use NIWI as an argument system for the language $(\mathcal{L}_{\mathbf{A}})^m$.

- $\mathsf{Setup}(1^\lambda, m)$ runs $\mathsf{pk}_{\mathsf{bind}} \xleftarrow{\$} \mathsf{HF.Setup}_{\mathsf{bind}}(1^\lambda)$ and $(\mathsf{crs}, \mathcal{T}) \xleftarrow{\$} \Pi.\mathsf{Gen}(1^\lambda)$ (for the language $\mathcal{L}_{\mathbf{A}}$ given by the matrix $\mathbf{A}$ defined in $\mathsf{pk}_{\mathsf{bind}}$), and outputs public parameters $\mathsf{pp} = (\mathsf{pk}_{\mathsf{bind}}, \mathsf{crs})$ and a trapdoor $\mathcal{T}$.
- $\mathsf{Stretch}(\mathsf{pp})$ samples $s = (s_1, \ldots, s_\lambda) \xleftarrow{\$} \{0,1\}^\lambda$ and $u_1, \ldots, u_\lambda \xleftarrow{\$} D_{\mathcal{U}}$, then computes $v_i = f_{\mathsf{pk}_{\mathsf{bind}}, s_i}(u_i)$, and finally outputs $\mathsf{pvk} = (v_i)_{i=1}^\lambda$, $x = G_m(s)$, and $\mathsf{aux} = (s, (u_i)_{i=1}^\lambda)$. Observe that the size of pvk does not depend on m.[6]
- $\mathsf{Prove}(\mathsf{pp}, \mathsf{aux}, i)$ (for $\mathsf{aux} = (s, (u_j)_{j=1}^\lambda)$) computes $v_i = f_{\mathsf{pk}_{\mathsf{bind}}, s_i}(u_i)$ exactly as $\mathsf{Stretch}$, and derives a witness $u^* = \mathsf{Eval}^{in}_{\mathsf{pk}_{\mathsf{bind}}}(G_{m,i}, (s_j, u_j)_{j=1}^\lambda)$ that explains $v^* = \mathsf{Eval}^{out}_{\mathsf{pk}_{\mathsf{bind}}}(G_{m,i}, (v_j)_{j=1}^\lambda)$ as $v^* = f_{\mathsf{pk}_{\mathsf{bind}}, b}(u^*)$. By our discussion above, we have that hence $v^* - b_i\mathbf{G} \in \mathcal{L}^m_{\mathbf{A}}$ with witness u^*. Hence, Prove next computes and outputs a proof $\pi \xleftarrow{\$} \Pi.\mathsf{Prove}(\mathsf{crs}, v^* - b\mathbf{G}, u^*)$.
- $\mathsf{Verify}(\mathsf{pp}, \mathsf{pvk}, \mathcal{T}, i, b, \pi)$ parses $\mathsf{pvk} = (v_i)_{i=1}^\lambda$, then computes

$$v^* = \mathsf{Eval}^{out}_{\mathsf{pk}_{\mathsf{bind}}}\left(G_{m,i}, (v_j)_{j=1}^\lambda\right),$$

  and finally returns $\Pi.\mathsf{Verify}(\mathsf{crs}, v^* - b_i\mathbf{G}, \pi, \mathcal{T})$.

Theorem 19. *Assume that LWE holds for the parameters from [22], that G_m is pseudorandom, and that Π is perfectly complete, computationally witness-indistinguishable and satisfies unbounded adaptive soundness. Then the above DVPRG is perfectly complete, equivocable, and has the unbounded binding property.*

We provide a proof of Theorem 19 in the full version [12].

References

1. Abusalah, H.: Generic instantiations of the hidden bits model for non-interactive zero-knowledge proofs for NP. Master's Thesis, RWTH Aachen (2013)
2. Benhamouda, F., Couteau, G., Pointcheval, D., Wee, H.: Implicit zero-knowledge arguments and applications to the malicious setting. In: Gennaro, R., Robshaw, M.J.B. (eds.) CRYPTO 2015, Part II. LNCS, vol. 9216, pp. 107–129. Springer, Heidelberg (Aug 2015)
3. Bitansky, N.: Verifiable random functions from non-interactive witness-indistinguishable proofs. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part II. LNCS, vol. 10678, pp. 567–594. Springer, Heidelberg (Nov 2017)
4. Bitansky, N., Paneth, O.: ZAPs and non-interactive witness indistinguishability from indistinguishability obfuscation. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 401–427. Springer, Heidelberg (Mar 2015)

[6] Strictly speaking, this may not be true, depending on G_m: the LWE parameters may depend on the size of the $G_{m,i}$, which in turn may depend on m. However, here we implicitly assume that $m \leq 2^\lambda$ and a suitable G_m (e.g., one in which $G_{m,i}(s)$ is of the form $F_s(i)$ for a PRF $F : \{0,1\}^\lambda \to \{0,1\}$).

5. Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications (extended abstract). In: 20th ACM STOC. pp. 103–112. ACM Press (May 1988)
6. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y.: Compressing vector OLE. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) ACM CCS 18. pp. 896–912. ACM Press (Oct 2018)
7. Canetti, R., Chen, Y., Reyzin, L., Rothblum, R.D.: Fiat-Shamir and correlation intractability from strong KDM-secure encryption. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part I. LNCS, vol. 10820, pp. 91–122. Springer, Heidelberg (Apr / May 2018)
8. Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 255–271. Springer, Heidelberg (May 2003)
9. Canetti, R., Lichtenberg, A.: Certifying trapdoor permutations, revisited. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018, Part I. LNCS, vol. 11239, pp. 476–506. Springer, Heidelberg (Nov 2018)
10. Cash, D., Kiltz, E., Shoup, V.: The twin Diffie-Hellman problem and applications. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 127–145. Springer, Heidelberg (Apr 2008)
11. Chaidos, P., Couteau, G.: Efficient designated-verifier non-interactive zero-knowledge proofs of knowledge. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part III. LNCS, vol. 10822, pp. 193–221. Springer, Heidelberg (Apr / May 2018)
12. Couteau, G., Hofheinz, D.: Designated-verifier pseudorandom generators, and their applications. Cryptology ePrint Archive (2019)
13. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (Apr / May 2002)
14. Damgård, I., Nielsen, J.B.: Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 581–596. Springer, Heidelberg (Aug 2002)
15. De Santis, A., Micali, S., Persiano, G.: Non-interactive zero-knowledge with preprocessing. In: Goldwasser, S. (ed.) CRYPTO'88. LNCS, vol. 403, pp. 269–282. Springer, Heidelberg (Aug 1990)
16. Dwork, C., Naor, M.: Zaps and their applications. In: 41st FOCS. pp. 283–293. IEEE Computer Society Press (Nov 2000)
17. Feige, U., Lapidot, D., Shamir, A.: Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In: 31st FOCS. pp. 308–317. IEEE Computer Society Press (Oct 1990)
18. Goldreich, O., Levin, L.A.: A hard-core predicate for all one-way functions. In: 21st ACM STOC. pp. 25–32. ACM Press (May 1989)
19. Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In: 27th FOCS. pp. 174–187. IEEE Computer Society Press (Oct 1986)
20. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM Journal on Computing 18(1), 186–208 (1989)
21. Goldwasser, S., Ostrovsky, R.: Invariant signatures and non-interactive zero-knowledge proofs are equivalent (extended abstract). In: Brickell, E.F. (ed.) CRYPTO'92. LNCS, vol. 740, pp. 228–245. Springer, Heidelberg (Aug 1993)

22. Gorbunov, S., Vaikuntanathan, V., Wichs, D.: Leveled fully homomorphic signatures from standard lattices. In: Servedio, R.A., Rubinfeld, R. (eds.) 47th ACM STOC. pp. 469–477. ACM Press (Jun 2015)
23. Goyal, R., Hohenberger, S., Koppula, V., Waters, B.: A generic approach to constructing and proving verifiable random functions. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part II. LNCS, vol. 10678, pp. 537–566. Springer, Heidelberg (Nov 2017)
24. Groth, J., Ostrovsky, R., Sahai, A.: Non-interactive zaps and new techniques for NIZK. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 97–111. Springer, Heidelberg (Aug 2006)
25. Groth, J., Ostrovsky, R., Sahai, A.: Perfect non-interactive zero knowledge for NP. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 339–358. Springer, Heidelberg (May / Jun 2006)
26. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (Apr 2008)
27. Katsumata, S., Nishimaki, R., Yamada, S., Yamakawa, T.: Designated verifier/prover and preprocessing NIZKs from diffie-hellman assumptions. Eurocrypt 2019 (2019)
28. Kilian, J., Petrank, E.: An efficient noninteractive zero-knowledge proof system for NP with general assumptions. Journal of Cryptology 11(1), 1–27 (Jan 1998)
29. Kim, S., Wu, D.J.: Multi-theorem preprocessing NIZKs from lattices. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 733–765. Springer, Heidelberg (Aug 2018)
30. Ong, S.J., Vadhan, S.P.: Zero knowledge and soundness are symmetric. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 187–209. Springer, Heidelberg (May 2007)
31. Oren, Y.: On the cunning power of cheating verifiers: Some observations about zero knowledge proofs (extended abstract). In: 28th FOCS. pp. 462–471. IEEE Computer Society Press (Oct 1987)
32. Ostrovsky, R., Wigderson, A.: One-way functions are essential for non-trivial zero-knowledge. In: Theory and Computing Systems, 1993., Proceedings of the 2nd Israel Symposium on the. pp. 3–17. IEEE (1993)
33. Pass, R., shelat, a., Vaikuntanathan, V.: Construction of a non-malleable encryption scheme from any semantically secure one. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 271–289. Springer, Heidelberg (Aug 2006)
34. Peikert, C., Vaikuntanathan, V.: Noninteractive statistical zero-knowledge proofs for lattice problems. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 536–553. Springer, Heidelberg (Aug 2008)
35. Quach, W., Rothblum, R.D., Wichs, D.: Reusable designated-verifier NIZKs for all NP from CDH. Eurocrypt 2019 (2019)
36. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) 37th ACM STOC. pp. 84–93. ACM Press (May 2005)
37. Rothblum, R.D., Sealfon, A., Sotiraki, K.: Towards non-interactive zero-knowledge for NP from LWE. PKC 2019 (2019)
38. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: Shmoys, D.B. (ed.) 46th ACM STOC. pp. 475–484. ACM Press (May / Jun 2014)
39. Vadhan, S.P.: An unconditional study of computational zero knowledge. In: 45th FOCS. pp. 176–185. IEEE Computer Society Press (Oct 2004)