Reusable Designated-Verifier NIZKs for all NP from CDH

Willy Quach
Northeastern University[*]

Ron D. Rothblum
Technion[†]

Daniel Wichs
Northeastern University[‡]

Abstract

Non-interactive zero-knowledge proofs (NIZKs) are a fundamental cryptographic primitive. Despite a long history of research, we only know how to construct NIZKs under a few select assumptions, such as the hardness of factoring or using bilinear maps. Notably, there are no known constructions based on either the computational or decisional Diffie-Hellman (CDH/DDH) assumption without relying on a bilinear map.

In this paper, we study a relaxation of NIZKs in the *designated verifier* setting (DV-NIZK), in which the public common-reference string is generated together with a secret key that is given to the verifier in order to verify proofs. In this setting, we distinguish between *one-time* and *reusable* schemes, depending on whether they can be used to prove only a single statement or arbitrarily many statements. For reusable schemes, the main difficulty is to ensure that soundness continues to hold even when the malicious prover learns whether various proofs are accepted or rejected by the verifier. One-time DV-NIZKs are known to exist for general NP statements assuming only public-key encryption. However, prior to this work, we did not have any construction of reusable DV-NIZKs for general NP statements from any assumption under which we didn't already also have standard NIZKs.

In this work, we construct reusable DV-NIZKs for general NP statements under the CDH assumption, without requiring a bilinear map. Our construction is based on the *hidden-bits paradigm*, which was previously used to construct standard NIZKs. We define a cryptographic primitive called a *hidden-bits generator (HBG)*, along with a designated-verifier variant (DV-HBG), which modularly abstract out how to use this paradigm to get both standard NIZKs and reusable DV-NIZKs. We construct a DV-HBG scheme under the CDH assumption by relying on techniques from the Cramer-Shoup hash-proof system, and this yields our reusable DV-NIZK for general NP statements under CDH.

We also consider a strengthening of DV-NIZKs to the *malicious designated-verifier* setting (MDV-NIZK) 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/verify proofs. We construct MDV-NIZKs under the "one-more CDH" assumption without relying on bilinear maps.

[*]Email: quach.w@husky.neu.edu.

[†]Email: rothblum@cs.technion.ac.il. Research supported in part by the Israeli Science Foundation (Grant No. 1262/18).

[‡]Email: wichs@ccs.neu.edu. Research supported by NSF grants CNS-1314722, CNS-1413964, CNS-1750795 and the Alfred P. Sloan Research Fellowship.

1 Introduction

(Non-Interactive) Zero-Knowledge. Zero-knowledge proofs, introduced in the seminal work of Goldwasser, Micali, and Rackoff [GMR85, GMR89], allow a prover to convince a verifier that a statement is valid without revealing anything beyond its validity. Standard zero-knowledge proof systems are interactive. Blum, Feldman, and Micali [BFM88] introduced the concept of non-interactive zero-knowledge (NIZK) proofs, which consist of a single message from the prover to the verifier. Such NIZKs cannot exist in the plain model, and are therefore considered in the *common reference string (CRS)* model, where a trusted third party chooses some common string (either uniformly at random or from some designated distribution) which is given to both the prover and the verifier. Such NIZKs for general **NP** statements have been constructed from a few select assumptions such as: (doubly-enhanced) trapdoor permutations which can be instantiated from factoring [BFM88, DMP88, FLS99, Gol11], the Diffie-Hellman assumption over bilinear groups [CHK03, GOS06] indistinguishability obfuscation [SW14] or fully exponential KDM hardness [CCRR18]. We also have such NIZKs in the random-oracle model [FS87].[1] However, despite a long history of research, we don't have any constructions based on several common standard assumptions: most notably the computational or decisional Diffie-Hellman assumptions (CDH, DDH) without requiring a bilinear map, or the learning-with-errors (LWE) assumption.

Designated-Verifier NIZK. In this work, we focus on a relaxed notion of NIZKs in the *designated-verifier* setting (DV-NIZK). In this model a trusted-third party generates a CRS together with secret key which is given to the verifier and is used to verify whether proofs are accepting or rejecting. We distinguish between schemes having *one-time* (a.k.a. single-theorem) security versus *reusable* (a.k.a. multi-theorem) security. One-time secure schemes only guarantee soundness for a single proof of a single statement. However, since the verifier's decision whether to accept or reject a proof depends on the secret key, a malicious prover may be able to learn something about the secret key over time by producing many proofs and seeing whether they are accepted or rejected by the verifier. Reusable DV-NIZKs ensure that soundness continues to hold even in such settings, where a prover can test whether the verifier accepts or rejects various proofs. In terms of constructions, there appears to be a huge gap between these notions. One-time secure DV-NIZKs were constructed for general **NP** statements assuming only the existence of public-key encryption [PsV06]. On the other hand, prior to this work, we did not have any constructions of reusable DV-NIZKs for general **NP** statements based on any assumptions under which we don't already also have standard NIZKs.

Malicious-Designated-Verifier NIZK. We also consider a strengthening of (reusable) DV-NIZKs to the malicious-designated-verifier setting (MDV-NIZKs). In this setting, the trusted party only generates a common uniformly random string. The verifier then gets to choose a public/secret key pair where the public key is used by the prover to generate proofs

[1]Additionally, we have constructions of NIZKs with an inefficient prover based on one-way permutations [FLS99]. In this work, we restrict ourselves to NIZKs where the prover can generate proofs efficiently given an **NP** witness.

and the secret key is used by the verifier to verify proofs. The main difference between DV-NIZKs and MDV-NIZKs is that, in the latter, we require zero-knowledge to hold even if the public key is chosen maliciously by the verifier. Therefore, an MDV-NIZK is similar to standard NIZKs in that the only *trusted setup* consists of a common random string, but an MDV-NIZK also requires additional *potentially untrusted setup* where the verifier publishes a public-key for which it keeps the corresponding secret key.

The notion of (reusable) MDV-NIZKs is equivalent to 2-round malicious-verifier ZK protocols in the common random string model (where the verifier's first-round message is reusable) by thinking of the verifier's public key as the first-round message. It is easy to see that the construction of *non-reusable* DV-NIZKs of [PsV06] extends naturally to yield *non-reusable* MDV-NIZKs assuming 2-round maliciously secure oblivious transfer in the common random string model. However, prior to this work, we did not have any constructions of *reusable* MDV-NIZKs for general **NP** statements based on any assumptions under which we don't already also have standard NIZKs.

Prior Work on DV-NIZKs and NIZKs with Pre-Processing. In prior work, the notion of DV-NIZKs was mainly studied in the context of non-malleable and CCA secure encryption. It is known that one-time DV-NIZKs allow us to compile any CPA secure (public-key) encryption scheme into a non-malleable one [PsV06] and reusable DV-NIZKs can compile it into a CCA secure one (by adapting the [NY90, DDN91] paradigm to the designated-verifier case). In this context, the work of Cramer and Shoup [CS98, CS02] constructed "hash-proof systems" which are unconditionally secure reusable DV-NIZKs for specific "algebraic" languages (e.g., the equality of two discrete logarithms) and used them to get practical CCA secure encryption. However, reusable DV-NIZKs have received surprisingly little attention as a general primitive. We believe that this notion is naturally interesting beyond its applications to non-malleable and CCA encryption. For example, it can take the place of standard NIZKs in the context of multiparty computation in scenarios where there is some (reusable) trusted setup.

DV-NIZKs can be thought of as a special case of a more general notion of "NIZKs with preprocessing" in which a trusted-third party creates a CRS together with two secret key: td_V given to the verifier and td_P given to the prover. We can consider two special cases of such NIZKs with preprocessing: if td_P is empty then this corresponds to the "designated-verifier" (DV-NIZK) setting that we study in this work, and if td_V is empty then we can think of this as a "designated-prover" (DP-NIZK). Several prior works study NIZKs with preprocessing [DMP90, KMO90, LS91, Dam93, DFN06, CC18] but all either (1) only consider specific "algebraic" languages rather than general **NP**, (2) are not reusable or (3) require assumptions such as factoring from which we already have standard NIZKs. The one exception is a very recent work of Kim and Wu [KW18] (CRYPTO 2018), which gave a novel construction of reusable DP-NIZKs for general **NP** languages under the LWE assumption. In that work, they explicitly asked the question whether one can construct reusable NIZKs in the preprocessing model under the CDH/DDH assumption. We answer their open question positively in this paper by constructing reusable DV-NIZKs under CDH. It remains a fascinating open question whether one can construct reusable DV-NIZKs under LWE, and conversely, whether one can construct reusable DP-NIZKs under CDH/DDH.

1.1 Our Results.

In this work, we construct reusable DV-NIZKs for general **NP** languages under the computational Diffie-Hellman (CDH) assumption without requiring a bilinear map.

Theorem 1.1. *Under the CDH assumption, there exists an (adaptively secure, statistically sound) reusable DV-NIZK proof system for all NP.*

We also construct reusable MDV-NIZKs for general **NP** languages under the one-more CDH (OM-CDH) assumption without requiring a bilinear map.

Theorem 1.2. *Under the One-More CDH assumption (Definition 6.3), there exists an (adaptively secure, statistically sound) reusable MDV-NIZK proof system for all NP.*

Our construction goes through the *hidden-bits* paradigm introduced by Feige, Lapidot and Shamir [FLS99] (see also [Gol01, Gol11]) to construct standard NIZKs. This paradigm consists of two steps. First, construct a NIZK for general **NP** statements in an idealized model called the "hidden-bits model" where the prover is given a long string of uniformly random bits and can choose to reveal some subset of them to the verifier. Such NIZKs in the hidden-bits model were constructed unconditionally with statistical soundness and zero knowledge. Second, use a cryptographic tool to compile NIZKs in the hidden-bits model to NIZKs in the CRS model. Such a compiler was constructed concretely using (doubly enhanced) trapdoor permutations, which can be instantiated based on factoring.

We generalize the second step of the hidden bits paradigm by defining a cryptographic primitive called a "hidden-bits generator" (HBG) which can be used to compile NIZKs in the hidden-bits model into ones in the CRS model.[2] This primitive modularizes the "hidden-bits paradigm" and simplifies the task of constructing NIZKs by reducing it to the task of constructing a HBG. We also clarify how to use HBG to get adaptive ZK security via the "hidden bits paradigm", which turns out to be surprisingly subtle and was not very clear from prior presentations of this paradigm. To get our main result, we generalize the hidden bits paradigm even further by extending the notion of HBG to the designated-verifier setting (DV-HBG) and the malicious-designated-verifier setting (MDV-HBG) and showing that the same compiler allows us to go from DV-HBG (resp. MDV-HBG) to reusable DV-NIZKs (resp. MDV-NIZKs). We then show how to construct DV-HBG from the computational Diffie-Hellman (CDH) assumption without bilinear maps. The last step uses the Cramer-Shoup hash-proof system, which can be thought of as a reusable DV-NIZK for equality of two discrete logarithms. Therefore we are in some sense bootstrapping a reusable DV-NIZK for this specific language to get a reusable DV-NIZK for all of **NP**. Finally, we show how to construct MDV-HBG from the one-more CDH (OM-CDH) assumption. This essentially starts with our construction of DV-HBG, which is clearly insecure in the the malicious-designated-verifier setting, and shows how to immunize it against malicious attacks. While the high level idea is simple, the proof of security is quite involved and uses techniques which may be of independent interest.

[2]A similar primitive called a "verifiable pseudorandom generator" was defined by [DN00] for the purpose of constructing ZAPs, which also lead to a construction of NIZKs.

1.2 Technical Overview

NIZKs via the Hidden-Bits Paradigm. We first review the "hidden-bits paradigm" proposed by [FLS99]; see [Gol01, Gol11] for a modern presentation which we follow here.

The starting point of this paradigm is a construction of NIZKs in an idealized model called the "hidden-bits model". In this model, there is a trusted third party that generates uniformly random bits $r_1, \dots, r_k$ and gives them to the prover. The prover outputs a proof π along with a subset $I \subseteq [k]$ of the bits to open. The verifier gets (I, π) from the prover together with the bits $\{r_i\}_{i \in I}$ from the trusted third party. Note that the verifier does not learn anything about the unopened bits $\{r_i\}_{i \notin I}$ and the prover cannot modify the values of the opened bits $\{r_i\}_{i \in I}$. Such NIZKs in the hidden-bits model can be constructed unconditionally with security against an unbounded prover/verifier where the soundness error can be made exponentially small.

The second step compiles NIZKs in the hidden-bits model into NIZKs in the CRS model. Such a compiler was presented by [FLS99, Gol01, Gol11] using doubly-enhanced trapdoor permutations (TDPs) (see also [BY93, GR13, CL17]). On a high level, the CRS consists of random values $y_1, \dots, y_k$ in the range of the TDP. The prover chooses a random permutation f_{com} along with an inversion trapdoor sk and inverts all of the values in the CRS to get preimages $x_1, \dots, x_k$. Define $r_1, \dots, r_k$ to be hardcore bits of $x_1, \dots, x_k$. The prover then runs the hidden-bits prover with $r_1, \dots, r_k$ to generate some proof (π, I) to which it appends the values $\mathsf{com}, \{x_i\}_{i \in I}$. The verifier checks $y_i = f_{\mathsf{com}}(x_i)$, computes $\{r_i\}_{i \in I}$ to be the hardcore bit of x_i and then runs the hidden bits verifier on (π, I). Intuitively, a malicious prover has a extremely limited ability to control the randomness $r_1, \dots, r_k$ by choosing com; by relying on an exponentially small soundness error of the hidden-bits proofs which survives a union-bound over all such com's, this flexibility is insufficient to break soundness. On the other hand, the verifier does not learn anything about the values $\{r_i\}_{i \notin I}$ by the security of the TDP.[3] While this is the high level approach, there are some subtleties involved; see [BY93, Gol01, Gol04, Gol11, GR13, CL17].

Hidden-Bits Generator (HBG). We begin by defining an abstract cryptographic primitive, which we call a hidden-bits generator (HBG), that can be used to compile NIZKs in the hidden-bits model to NIZKs in the CRS model. An HBG that generates k bits consists of three algorithms:

- Setup creates a crs.
- $\mathsf{GenBits}(\mathsf{crs})$ outputs a *short* commitment com whose size is much smaller than k, along with hidden-bits $\{r_i\}_{i \in [k]}$, and certificates $\{\pi_i\}_{i \in [k]}$.
- $\mathsf{Verify}(\mathsf{crs}, \mathsf{com}, i, r_i, \pi_i)$ checks the certificate π_i to verify that r_i is indeed the i'th hidden bit.

An HBG should satisfy two simple properties. Firstly, we require the scheme to be *statistically binding*, meaning that $(\mathsf{crs}, \mathsf{com})$ together completely determine some sequence of bits $r_1, \dots, r_k$ and no (even inefficient) prover can come up with a valid certificate π'_i

[3]The basic compiler only achieves zero-knowledge for a single theorem and [FLS99] then relies on another generic compiler via the "or trick" to go from single-theorem to multi-theorem zero-knowledge.

for the wrong bit $r'_i \neq r_i$. Intuitively, by combining the above property together with the requirement that com is short, we ensure that the prover does not have much control over the bits r_i that he can open and the limited control that he does have is insufficient to break the soundness of the hidden-bits NIZK (by amplifying its soundness sufficiently so that it survives a union bound over all the com's that the prover can choose). Secondly, we require the scheme to be *computationally hiding*, meaning that for any set $I \subseteq [k]$, if we are given honestly generated $\mathsf{crs}, \mathsf{com}, \{r_i, \pi_i\}_{i \in I}$ then the "unopened" hidden bits $\{r_j\}_{j \notin I}$ are computationally indistinguishable from uniform.

Compiling from Hidden-Bits Model to CRS Model. Intuitively, we would like to use HBG to compile NIZKs from the hidden-bits model to the CRS model by letting the prover generate the hidden-bits via the HBG $\mathsf{GenBits}$ algorithm. There are two issues with this basic approach:

- For soundness, if the malicious prover chooses a "bad" (not uniformly random) com then the HBG abstraction does not provide any guarantees that the bits r_i to which he is committed are random and hence we cannot rely on the soundness of the hidden-bits NIZK.
- For zero-knowledge, we notice that the honest hidden-bits prover may choose the set I adaptively depending on all of the bits $\{r_i\}_{i \in [k]}$ (and indeed this is the case for the hidden-bits NIZK constructed in [FLS99]) and we still need to argue that the unopened bits $\{r_j\}_{j \notin I}$ are hidden. The hiding property of HBG only guarantees that the unopened bits are hidden when I is chosen ahead of time.

To fix both of the above issues we add additional uniformly random bits $s_1, \ldots, s_k$ to the CRS of the NIZK and define the hidden-bits to be $r_i \oplus s_i$ where r_i comes from the HBG. This ensures that for any fixed com chosen by a malicious prover the hidden-bits that he can open are uniform over the choice of s_i.[4] It also ensures that the choice of the set I chosen by the honest hidden-bits prover is independent of the outputs r_i of the HBG and therefore allows us to rely on HBG security.

We uncover an additional complication when proving *adaptive* ZK, where the malicious verifier can choose the statement to be proven adaptively depending on the CRS. The work of [FLS99] showed adaptive ZK for their particular protocol (using particular hidden-bits NIZK) but it did not give a modular proof. Indeed, our attempts to prove that the compiler can generically start with any hidden-bits NIZK and achieve adaptive ZK failed for subtle reasons involving "selective opening" failures. Instead, we were able to abstract out a special property of the hidden-bits NIZK of [FLS99] which we call "special ZK", which we show to be sufficient to get adaptive ZK in the CRS model via the above compiler.

Using the compiler, we reduce the task of constructing NIZKs to that of constructing an HBG, which is a conceptually much simpler primitive.

Designated Verifier Setting: (M)DV-HBG to (M)DV-NIZK. We generalize the notion of HBG to the designated-verifier setting (DV-HBG). The only differences are that:

[4]The fact that the prover can adaptively choose com *after* seeing $s_1, \ldots, s_k$ is handled by simply taking a union bound over all possible choices of com.

(1) the Setup algorithm generates a crs together with a trapdoor td which is given to the verifier and the Verify algorithm takes the trapdoor td as an input, (2) we modify the statistically binding security property to hold even if a computationally unbounded prover can make polynomially many queries to the $\mathsf{Verify}(\mathsf{crs}, \mathsf{td}, \cdots)$ oracle which allows it to check whether various certificates are valid or invalid, and (3) we modify the computationally hiding property to hold even given td. To get our main result, we naturally extend our compiler to show that DV-HBG allows us to compile NIZKs in the hidden-bits model into reusable DV-NIZKs. Therefore, we reduce the task of constructing reusable DV-NIZKs to that of constructing DV-HBG.

We further generalize the notion of HBG to the malicious-designated-verifier setting (MDV-HBG). Now, in addition to a Setup algorithm that generates the crs there is a KeyGen algorithm that generates a public key pk along an associated secret key sk. Essentially, we think of $\mathsf{crs}, \mathsf{pk}$ as together corresponding to the crs in the previous definition, and of sk as the trapdoor. The binding property is essentially the same as before. However, we require that hiding holds even if pk is generated maliciously (and adaptively depending on crs). We show that MDV-HBG allows us to compile NIZKs in the hidden-bits model into reusable MDV-NIZKs. Therefore, we reduce the task of constructing reusable MDV-NIZKs to that of constructing MDV-HBG.

DV-HBG from CDH. We show how to instantiate a designated-verifier DV-HBG based on the computational Diffie-Hellman (CDH) assumption to get our reusable DV-NIZK from CDH. Our construction relies on the ideas underlying the Cramer-Shoup (1-universal) hash-proof system [CS98, CS02] which can be thought of as an unconditionally secure reusable DV-NIZK for the "equality of two discrete logs" – i.e., given some public group elements g, h we define the language consisting of tuples (g', h') such that $\mathsf{DLOG}_g(g') = \mathsf{DLOG}_h(h')$. In particular, we think of the projection key of the hash-proof system as the CRS of the DV-NIZK, and the hashing key as the associated trapdoor. In the body of our paper, we give our full construction using the specific Cramer-Shoup instantiation, but for the introduction we will treat the Cramer-Shoup reusable DV-NIZK proof system as a black-box.

Our DV-HBG construction works as follows. Let $\mathbb{G}$ be some cyclic group of order p and let g be a generator.

- The Setup algorithm chooses random group elements $h_1, \ldots, h_k$. It also instantiates k copies of the Cramer-Shoup DV-NIZK with respect to the public group elements (g, h_i) respectively. The crs consists of $g, h_1, \ldots, h_k$ together with the k values $\{\mathsf{crs}_i\}_{i\in[k]}$ of the Cramer-Shoup DV-NIZK. The trapdoor $\mathsf{td} = \{\mathsf{td}_i\}_{i\in[k]}$ consists of the k trapdoors for the Cramer-Shoup DV-NIZK.

- The $\mathsf{GenBits}(\mathsf{crs})$ algorithm chooses $y \leftarrow \mathbb{Z}_q$ and sets $\mathsf{com} = g^y$. For $i = 1 \ldots, k$, it sets $t_i = h_i^y$, $r_i = \mathsf{hc}(t_i)$, where hc is a hardcore predicate (e.g., Goldreich-Levin [GL89]). Finally it sets $\pi_i = (t_i, \pi_i^{CS})$ where π_i^{CS} is a Cramer-Shoup proof that $\mathsf{DLOG}_g(\mathsf{com}) = \mathsf{DLOG}_{h_i}(t_i)$.

- The Verify algorithm gets r_i and $\pi_i = (t_i, \pi_i^{CS})$ and checks that $r_i = \mathsf{hc}(t_i)$ and that π_i^{CS} is a valid Cramer-Shoup proof using the corresponding trapdoor td_i.

For the statistically binding property we note that given $\mathsf{crs}, \mathsf{com}$ the values $t_i = h_i^y$ and therefore also the hidden bits $r_i = \mathsf{hc}(t_i)$ are completely determined. The prover cannot lie about t_i and therefore also about r_i by the unconditional reusable security of the Cramer-Shoup proof, and this holds even given oracle access to the Cramer-Shoup verifier. For the computational hiding property we rely on the fact that, given g, h_i, g^y, the CDH assumption ensures that h_i^y is computationally unpredictable and therefore $\mathsf{hc}(h_i^y)$ is indistinguishable from uniform. This holds even given h_j, h_j^y for various random h_j since the distinguisher can sample such values himself by sampling $h_j = g^{x_j}$ and computing $h_j^y = (g^y)^{x_j}$.

MDV-HBG from One-More CDH. Finally, we show how to instantiate our malicious-designated-verifier MDV-HBG based on the one-more CDH assumption to get our reusable MDV-NIZK from one-more CDH. The construction and the security intuition are somewhat involved and so we present them in several stages.

Initial Attempt. As a first attempt, we can try to use the previous construction directly as an MDV-HBG. In particular, we can set the crs to only consist of the uniformly random values $\mathsf{crs} = (h_1, \dots, h_k)$. The Cramer-Shoup DV-NIZKs then naturally define pk, sk. Here it helps to be concrete about how the Cramer-Shoup DV-NIZK works. For each i, the Cramer-Shoup proof system defines $\mathsf{pk}_i = h_i^{a_i} g^{b_i}$ and the corresponding $\mathsf{sk}_i = (a_i, b_i)$. The MDV-HBG public keys and secret keys consist of these values $\mathsf{pk} = \{\mathsf{pk}_i\}, \mathsf{sk} = \{\mathsf{sk}_i\}$. Given a commitment $\mathsf{com} = g^y$, recall that the i'th hidden bit is defined by taking a hardcore predicate $r_i = \mathsf{hc}(t_i)$ where $t_i = h_i^y$. The opening to the i'th hidden bit consists of $t_i = h_i^y$ and the Cramer-Shoup proof $\pi_i^{CS} = \mathsf{pk}_i^y$.

Attack On Initial Attempt. Unfortunately, it's clear that the above is not secure as MDV-HBG. For example, if the malicious verifier chooses $\mathsf{pk}_i = h_j$ for $j \neq i$ then, by opening the i'th hidden bit and giving a proof $\pi_i^{CS} = \mathsf{pk}_i^y = h_j^y$, the prover inadvertently also reveals the j'th hidden bit! While the above is easily detectable, the malicious verifier can alternately set $\mathsf{pk}_i = h_j^x$ for a random x and still perform the same attack without being detectable. At the very least, we need to modify our solution to overcome this particular attack.

The Fix. We start with the above "base scheme", which is not secure in the MDV setting, and show how to immunize against the above attack. To do so, we use the "base scheme" to generate ℓ "base hidden values" for some $\ell \gg k$ and then combine them carefully to create the k "actual hidden bits". Recall that the base scheme defines a commitment g^y and the ℓ base hidden values are $t_j = h_j^y$. We can open any base value by giving the opening $\pi_j^{CS} = \mathsf{pk}_j^y$.

Instead of using the base values directly, we define each of the k "actual hidden bits" by combining together a small group of base values and applying a (Goldreich-Levin) hardcore predicate hc. The groups are chosen via a pseudo-random mapping φ which maps each $i \in [k]$ to a small group $\varphi(i) \subseteq [\ell]$. In other words, the i'th actual hidden bit is defined as $r_i = \mathsf{hc}(\{t_j \,:\, j \in \varphi(i)\})$. The mapping φ is chosen by the prover and is a part of com. To open any actual hidden bit $i \in [k]$ the prover opens all of the base hidden values t_j^y and also provides the corresponding Cramer-Shoup proofs pk_j^y for $j \in \varphi(i)$. Note that, since φ is a part of com and we require com to be short, it is important that φ has a short description size and therefore it must be a pseudo-random rather than truly random mapping. For concreteness, we set the number of based hidden values to $\ell = 3k\lambda$ and the group size to

$|\varphi(i)| = \lambda$, where λ is the security parameter.

Intuition for the Fix. Intuitively, this prevents the above attacks for the following reason. Assume that the verifier can choose pk maliciously so that the opening of any base value j can inadvertently also reveal some other base value $j' = \psi(j)$, where ψ is some mapping defined implicitly by the choice of pk. Nevertheless, it is likely that each hidden bit i depends on some hidden value $j \in \varphi(i)$ that is not revealed even if we open all the other hidden bits $i' \neq i$. In particular, opening the bits $i' \neq i$ corresponds to giving out the base hidden value j' as well as the inadvertently opened values $\psi(j')$ for each $j' \in \varphi(i')$. But the entire set of revealed values $R = \{j', \psi(j') \ : \ j' \in \varphi(i'), i' \neq i\}$ is of size $|R| \leq 2k\lambda$ and $\varphi(i) \subset [\ell = 3k\lambda]$ appears to be a random and independent subset of size $|\varphi(i)| = \lambda$. Hence it is likely that $\varphi(i)$ contains some value $j \notin R$ which was not revealed. Here we crucially rely on the fact that φ is chosen (pseudo-)randomly by the prover after the verifier chooses pk which defines the mapping ψ.

The One-More CDH Assumption. While the above idea seems to immunize against the particular class of attacks we previously discussed, proving security against general attacks is more challenging. Nevertheless, we manage to do so under the "one-more CDH" assumption. The one-more CDH assumption considers an adversary who is given $g, g^y, h_1, \ldots, h_k$ along with an oracle $O_y(\cdot)$ which takes as input an arbitrary group element f and returns $O_y(f) = f^y$. It says that even if the adversary makes m arbitrary calls to the oracle O_y he cannot predict more than m of the values $\{h_j^y\}$.

Security Under One-More CDH. Our high level proof goes as follows. Assume that a malicious verifier gets to choose $\mathsf{pk} = \{\mathsf{pk}_j\}_{j\in[\ell]}$ maliciously after seeing $\mathsf{crs} = \{h_j\}_{j\in[\ell]}$ and can break hiding. This means that for some $i \in [k]$, if the verifier gets a random com and openings to all the hidden bits *except* for the i'th one, he can distinguish hidden bit i from uniform with non-negligible advantage. Since the i'th hidden bit is defined by taking the Goldreich-Levin hardcore bit of the base hidden values $j \in \varphi(i)$, this means that the verifier can also predict all these values with non-negligible probability. So, if the verifier gets $\varphi, g, g^y, \{h_j^y, \mathsf{pk}_j^y \ : \ j \in \varphi(i'), i' \in [k] \setminus \{i\}\}$ then he can predict $\{h_j^y : j \in \varphi(i)\}$. Intuitively, we want to use such a verifier to break one-more CDH.

But in the above scenario, the verifier gets many more values raised to the y power than he is able to output. To get around this, we want to "rewind" the verifier run him on many different choices of φ to get more values $\{h_j^y : j \in \varphi(i)\}$ out of him. But each time we rewind we also need to provide him with the appropriate values $\{h_j^y, \mathsf{pk}_j^y \ : \ j \in \varphi(i'), i' \in [k] \setminus \{i\}\}$ so we are again getting fewer powers of y out than we need to put in, which appears to be self-defeating. If φ were truly random, we could get around this by freshly sample $\varphi(i)$ on each rewinding but keep $\varphi(i') \ : \ i' \in [k] \setminus i$ fixed – that way we would only need to give out some fixed $2k\lambda$ values $\{h_j^y, \mathsf{pk}_j^y \ : \ j \in \varphi(i'), i' \in [k] \setminus \{i\}\}$ but on each rewinding we get some additional fresh values $\{h_j^y : j \in \varphi(i)\}$ out of the verifier, and eventually we get more out than we put in which allows us to break one-more CDH.

Unfortunately φ needs to have a short description, and therefore can only be pseudorandom, in which case it's not clear how to freshly re-sample $\varphi(i)$ while keeping $\varphi(i') \ : \ i' \in [k] \setminus i$ fixed. We resolve this issue by using a special form of pseudorandom functions (PRFs) called "somewhere equivocal PRFs" [HJO+16] which essentially allow us to do exactly this while keeping the description of φ short. Furthermore, such somewhere equivocal PRFs were constructed from only one-way functions using the ideas of "distributed point functions"

[GI14, BGI15] and therefore don't introduce any additional assumptions.

1.3 Concurrent works

Concurrently and independently of ours, the works of [CH19] and [KNYY19] present a similar construction of reusable DV-NIZKs from CDH, compiling the hidden-bits NIZK of [FLS99] using the Cramer-Shoup hash-proof system [CS98, CS02, CKS08]. Additionally, they respectively obtain the following results:

- [CH19] gives a construction of NIZKs for all NP assuming LWE, along with a *non-interactive witness intistinguishable* (NIWI) proof for the Bounded Distance Decoding problem.
- [KNYY19] builds pre-processing NIZKs for all NP with *succinct* proofs, namely a pre-processing NIZK from DDH with proofs of size $|C| + \mathsf{poly}(\lambda)$ (where C is a circuit checking the NP relation), and a designated-prover NIZK from (strong) assumptions over pairing-friendly groups, with proof size $|C| + \mathsf{poly}(\lambda)$.

Meanwhile, our work introduces the notion of *malicious designated-verifier* NIZKs (MDV-NIZK), and presents a construction from the One-More CDH assumption.

Organization

Basic definitions and notations are given in Section 2. In Section 3 we introduce our new notion of Hidden Bits Generator (HBG). In Section 4 we show how to use an HBG to construct NIZKs. In Section 5 we construct a designated-verifier Hidden Bits Generator assuming CDH. A few extension are mentioned in Section 7.

Lastly, in Appendix A we give a construction of a HBG from the CDH assumption over bilinear groups and in Appendix B we construct a HBG from (doubly-enhanced) trapdoor permutations.

2 Preliminaries

We will denote by λ the security parameter. The notation $\mathsf{negl}(\lambda)$ denotes any function f such that $f(\lambda) = \lambda^{-\omega(1)}$, and $\mathsf{poly}(\lambda)$ denotes any function f such that $f(\lambda) = \mathcal{O}(\lambda^c)$ for some $c > 0$.

We define the statistical distance between two random variables X and Y over some domain Ω as: $\mathbf{SD}(X, Y) = \frac{1}{2}\sum_{w\in\Omega} |X(w) - Y(w)|$. We say that two ensembles of random variables $X = \{X_\lambda\}$, $Y = \{Y_\lambda\}$ are *statistically indistinguishable*, denoted $X \overset{s}{\approx} Y$, if $\mathbf{SD}(X_\lambda, Y_\lambda) \leq \mathsf{negl}(\lambda)$.

We say that two ensembles of random variables $X = \{X_\lambda\}$, and $Y = \{Y_\lambda\}$ are *computationally indistinguishable*, denoted $X \overset{c}{\approx} Y$, if, for all (non-uniform) PPT distinguishers Adv, we have $|\Pr[\mathsf{Adv}(X_\lambda) = 1] - \Pr[\mathsf{Adv}(Y_\lambda) = 1]| \leq \mathsf{negl}(\lambda)$.

For a set X, integer k and sequence $x \in X^k$, we denote by x_i the i-th entry in the sequence, for any $i \in [k]$. For a subset $I \subset [k]$, we denote by $x_I = (x_i)_{i\in I}$ the subsequence of x in locations I.

For a probabilistic algorithm $\mathsf{alg}(\cdot)$, we may explicit its internal randomness as follows: $\mathsf{alg}(\,\cdot\,;\mathsf{coins})$.

2.1 The Diffie-Hellman Assumption

A *group generator* $(\mathbb{G}, p, g) \leftarrow \mathsf{GroupGen}(1^\lambda)$ is a PPT algorithm which, on input 1^λ, outputs the description of a cyclic group $\mathbb{G}$ of order p, and a generator g of $\mathbb{G}$. We require that there are efficient algorithms running in time $\mathsf{poly}(\lambda)$ to perform the group operation in $\mathbb{G}$ and to test membership in $\mathbb{G}$. For notational simplicity, we will often shorten such an output $(\mathbb{G}, p, g)$ to $\mathbb{G}$ and assume that g, p are implicit. A *prime-order group generator* additionally ensures that p is prime.

Definition 2.1 (Computational Diffie-Hellman (CDH) assumption)**.** *Let* $\mathsf{GroupGen}$ *be a group generator. We say that the* Computational Diffie-Hellman (CDH) assumption *holds relative to* $\mathsf{GroupGen}$ *if for all PPT algorithm $\mathcal{A}$, we have:*

$$\Pr\left[\mathcal{A}\left(\mathbb{G}, p, g, g^a, g^b\right) = g^{ab} \;\;:\;\; (\mathbb{G}, p, g) \leftarrow \mathsf{GroupGen}(1^\lambda),\; (a, b) \overset{\$}{\leftarrow} \mathbb{Z}_p^2\right] \leq \mathsf{negl}(\lambda).$$

Given such a group generator satisfying the CDH assumption, we can consider an associated (randomized) *hard-core bit* $\mathsf{hc} : \mathbb{G} \to \{0, 1\}$ such that for all PPT algorithm $\mathcal{A}$, we have:

$$\Pr\left[\mathcal{A}\left(\mathbb{G}, p, g, g^a, g^b, \tau\right) = \mathsf{hc}(g^{ab}\;;\;\tau) \;\;:\;\; \begin{array}{rl} \tau & \overset{\$}{\leftarrow} \{0,1\}^{L(\lambda)} \\ (\mathbb{G}, p, g) & \leftarrow \mathsf{GroupGen}(1^\lambda) \\ (a, b) & \overset{\$}{\leftarrow} \mathbb{Z}_p^2 \end{array}\right] \leq 1/2 + \mathsf{negl}(\lambda),$$

where the hard-core bit hc uses $L(\lambda)$ random coins.

Such a hard-core bit can be generically obtained, using the Goldreich-Levin construction [GL89].

2.2 Reusable Designated-Verifier NIZKs

In this section we define the notion of Reusable Designated-Verifier NIZKs (and obtain the standard notion of NIZK as a special case).

Definition 2.2 (Reusable DV-NIZKs)**.** *Let be L an NP language with witness relation R_L. A* Reusable Designated-Verifier Non-Interactive Zero-Knowledge (DV-NIZK) Proof *for L is a tuple of PPT algorithms* $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ *where:*

- $\mathsf{Setup}(1^\lambda, 1^n)$*: On input the security parameter λ and statement length n, outputs a common reference string* crs *and a trapdoor* td*;*
- $\mathcal{P}(\mathsf{crs}, x, w)$*: On input a common reference string* crs*, a statement x of length n and a witness w, outputs a proof π;*
- $\mathcal{V}(\mathsf{crs}, \mathsf{td}, x, \pi)$*: On input a common reference string* crs*, a trapdoor* td*, a statement x and a proof π, outputs* `accept` *or* `reject`*,*

such that they satisfy the following properties:

- ***Completeness**: We require that for all $(x,w) \in R_L$, we have:*

$$\Pr\left[\mathcal{V}(\mathsf{crs},\mathsf{td},x,\pi) = \mathtt{accept} \ : \ \begin{array}{cl} (\mathsf{crs},\mathsf{td}) & \leftarrow \mathsf{Setup}(1^\lambda, 1^{|x|}) \\ \pi & \leftarrow \mathcal{P}(\mathsf{crs},x,w) \end{array}\right] = 1;$$

- ***Statistical Soundness**: Let n and Q be any polynomials, and let $\widetilde{\mathcal{P}}$ be any (computationally unbounded) cheating prover that makes at most $Q(\lambda)$ queries to an oracle $\mathcal{V}(\mathsf{crs},\mathsf{td},\cdot,\cdot)$ which takes as input (x,π), and outputs $\mathcal{V}(\mathsf{crs},\mathsf{td},x,\pi)$). We require that:*

$$\Pr\left[\mathcal{V}(\mathsf{crs},\mathsf{td},x,\pi) = \mathtt{accept} \wedge x \notin L \ : \ \begin{array}{cl} (\mathsf{crs},\mathsf{td}) & \leftarrow \mathsf{Setup}(1^\lambda, 1^{n(\lambda)}) \\ (x,\pi) & \leftarrow \widetilde{\mathcal{P}}^{\ \mathcal{V}(\mathsf{crs},\mathsf{td},\cdot,\cdot)}(\mathsf{crs}) \end{array}\right] \leq \mathsf{negl}(\lambda);$$

- ***Zero-Knowledge (Selective)**: We require that there exists a PPT simulator* Sim *such that for any PPT stateful[5] adversary $\mathcal{A}$, the two following distributions are computationally indistinguishable:*

$\text{EXP}^{\mathit{Real}}(1^\lambda)$:	$\text{EXP}^{\mathit{Ideal}}(1^\lambda)$:
$(x,w) \leftarrow \mathcal{A}(1^\lambda)$	$(x,w) \leftarrow \mathcal{A}(1^\lambda)$
where $(x,w) \in R_L$	*where* $(x,w) \in R_L$
$(\mathsf{crs},\mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda, 1^{\lvert x \rvert})$, $\pi \leftarrow \mathcal{P}(\mathsf{crs},x,w)$	$(\mathsf{crs},\mathsf{td},\pi) \leftarrow \mathsf{Sim}(1^\lambda, x)$
Output $\mathcal{A}(\mathsf{crs},\mathsf{td},\pi)$	*Output* $\mathcal{A}(\mathsf{crs},\mathsf{td},\pi)$

Our basic definition only considers selective ZK where the statement being proven is chosen ahead of time, prior to seeing the CRS. In Section 4.2 we also consider a stronger notion of adaptive ZK.

Our definition of designated-verifier NIZK coincides with that of standard (publicly verifiable) NIZK if the trapdoor td is empty.

Definition 2.3. *A* publicly-verifiable NIZK *is a reusable designated-verifier NIZK where the trapdoor* td *output by* Setup *is an empty string.*

Remark 2.4 (Bounding the number of queries to the Verify oracle)**.** *Notice that for soundness we only allow the unbounded cheating prover to make a polynomial number of queries to $\mathcal{V}(\mathsf{crs},\mathsf{td},\cdot,\cdot)$. One would ideally allow the unbounded cheating prover to make* arbitrarily *many queries to $\mathcal{V}$ (matching more closely the publicly-verifiable setting, where a cheating prover can indeed query the verification algorithm on arbitrarily many inputs). It turns out that any DV-NIZK satisfying this stronger notion can be generically turned into a publicly-verifiable one. This is because the cheating prover can query all possible proofs to $\mathcal{V}$ for any $x \notin L$; and therefore soundness can only hold if there are no valid proof of any false statement (with overwhelming probability over the choice of* crs*), in which case soundness also holds when the prover is given the trapdoor. Therefore this is essentially the best requirement one can hope for as a meaningful notion of reusable DV-NIZKs which is weaker than publicly-verifiable ones.*

[5]Throughout this paper we follow the convention that whenever a stateful adversary $\mathcal{A}$ is invoked with some inputs it also produces some state which it gets as input on the next invocation.

Remark 2.5 (Single-Theorem vs. Multi-Theorem Zero-Knowledge.)**.** *The definition of ZK above is often referred to as "single-theorem ZK" since it only requires zero-knowledge to hold for a* single *statement. However, there is a generic compiler from single-theorem ZK to multi-theorem ZK where zero-knowledge holds polynomially many statements via the "OR trick" [FLS99]. We note that the very same transformation directly applies to both the selective and adaptive ZK setting and also both the publicly-verifiable and the designated-verifier setting.*

2.3 NIZKs in the Hidden-Bits Model

We now recall the definition of a NIZK in the hidden-bits model:

Definition 2.6 (NIZK in the Hidden-Bits Model)**.** *Let L be an NP language and n be an integer. A* Non-Interactive Zero-Knowledge Proof in the Hidden-Bits Model *for L is given by a pair of PPT algorithms $(\mathcal{P}, \mathcal{V})$, and a polynomial $k(\lambda, n)$, where:*

- $\mathcal{P}(1^\lambda, r, x, w)$*: On input string $r \in \{0,1\}^{k(\lambda,n)}$, a statement x of size $|x| = n$ and a witness w, output a set of indices $I \subseteq [k]$ and proof π.*
- $\mathcal{V}(1^\lambda, I, r_I, x, \pi)$*: On input a subset $I \subseteq [k]$, a string r_I, a statement x and a proof π, outputs* `accept` *or* `reject`*,*

such that they satisfy the following properties:

- ***Completeness****: We require that for all $x \in L$ of size $|x| = n$ with witness w we have:*

$$\Pr\left[\mathcal{V}(1^\lambda, I, r_I, x, \pi) = \texttt{accept} \;:\; \begin{array}{cl} r & \xleftarrow{\$} \{0,1\}^{k(\lambda,n)} \\ (I, \pi) & \leftarrow \mathcal{P}(1^\lambda, r, x, w) \end{array}\right] = 1;$$

- ***Soundness****: We require that for all polynomial $n = n(\lambda)$, and all unbounded cheating prover $\widetilde{\mathcal{P}}$, we have:*

$$\Pr\left[\begin{array}{cl} & \mathcal{V}(1^\lambda, I, r_I, x, \pi) = \texttt{accept} \\ \wedge & x \notin L \\ \wedge & |x| = n \end{array} \;:\; \begin{array}{cl} r & \xleftarrow{\$} \{0,1\}^{k(\lambda,n)} \\ (x, \pi, I) & \leftarrow \widetilde{\mathcal{P}}(1^\lambda, r) \end{array}\right] \leq \mathsf{negl}(\lambda);$$

- ***Zero-Knowledge****: We require that there exists an efficient simulator* Sim *such that for any adversary $\mathcal{A}$ the two following distributions are statistically indistinguishable:*

$$(I, r_I, \pi) \overset{s}{\approx} (I', r'_I, \pi')$$

where $(x, w) \leftarrow \mathcal{A}(1^\lambda), r \leftarrow \{0,1\}^{k(\lambda,|x|)}, (I, \pi) \leftarrow \mathcal{P}(1^\lambda, r, x, w), (I', r'_I, \pi') \leftarrow \mathsf{Sim}(1^\lambda, x)$.

When clear from context, we will omit 1^λ as an argument to the algorithms defined above.

Remark 2.7 (Amplifying soundness)**.** *Let $\ell(\lambda, n)$ be a polynomial. Then, given any NIZK in the hidden-bits model, we can build one with soundness $2^{-\ell(\lambda,n)} \cdot \mathsf{negl}(\lambda)$. This is simply done by running $\ell(\lambda, n)$ copies of the NIZK in parallel, and where the new verification algorithm accepts a proof if and only if all of the executions accept. Note that doing so requires to use $k \cdot \ell(\lambda, n)$ hidden bits instead of k initially.*

Theorem 2.8 ([FLS99], see also [Gol01, Section 4.10.2])**.** *Every $L \in \mathbf{NP}$ has a NIZK in the Hidden-Bits Model.*

3 Hidden-Bits Generator

In this section, we define our new notion of Hidden-Bits Generator (HBG). For simplicity, we first define a publicly verifiable version of HBG and then extend the definition to a designated-verifier version (DV-HBG).

Definition 3.1 (Hidden-Bits Generator)**.** *A* Hidden-Bits Generator (HBG) *is given by a set of PPT algorithms* $(\mathsf{Setup}, \mathsf{GenBits}, \mathsf{Verify})$*:*

- $\mathsf{Setup}(1^\lambda, 1^k)$*: Outputs a common reference string* crs*.*
- $\mathsf{GenBits}(\mathsf{crs})$*: Outputs a triple* $\left(\mathsf{com}, r, \{\pi_i\}_{i\in[k]}\right)$*, where* $r \in \{0,1\}^k$*.*
- $\mathsf{Verify}(\mathsf{crs}, \mathsf{com}, i, r_i, \pi_i)$*: Outputs* `accept` *or* `reject`*, where* $i \in [k]$*.*

We require any Hidden-Bits Generator to satisfy the following properties:

Correctness: *We require that for every polynomial $k = k(\lambda)$ and for all $i \in [k]$, we have:*

$$\Pr\left[\mathsf{Verify}(\mathsf{crs}, \mathsf{com}, i, r_i, \pi_i) = \mathtt{accept} \;:\; \begin{array}{cl} \mathsf{crs} & \leftarrow \mathsf{Setup}(1^\lambda, 1^k) \\ (\mathsf{com}, r, \pi_{[k]}) & \leftarrow \mathsf{GenBits}(\mathsf{crs}) \end{array}\right] = 1.$$

Succinct Commitment: *We require that there exists some set $\mathcal{COM}(\lambda)$ and some constant $\delta < 1$ such that $|\mathcal{COM}(\lambda)| \le 2^{k^\delta \mathsf{poly}(\lambda)}$, and such that for all* crs *output by* $\mathsf{Setup}(1^\lambda, 1^k)$ *and all* com *output by* $\mathsf{GenBits}(\mathsf{crs})$ *we have* $\mathsf{com} \in \mathcal{COM}(\lambda)$*. Furthermore, we require that for all* $\mathsf{com} \notin \mathcal{COM}(\lambda)$*,* $\mathsf{Verify}(\mathsf{crs}, \mathsf{com}, \cdot, \cdot)$ *always outputs* `reject`*.*[6]

Statistical Binding: *There exists an (inefficient) deterministic algorithm* $\mathsf{Open}(1^k, \mathsf{crs}, \mathsf{com})$ *such that for every polynomial $k = k(\lambda)$, on input 1^k,* crs *and* com*, the algorithms outputs r such that for every (potentially unbounded) cheating prover $\widetilde{\mathcal{P}}$:*

$$\Pr\left[\begin{array}{cl} & r_i^* \neq r_i \\ \wedge & \mathsf{Verify}(\mathsf{crs}, \mathsf{com}, i, r_i^*, \pi_i) = \mathtt{accept} \end{array} \;:\; \begin{array}{cl} \mathsf{crs} & \leftarrow \mathsf{Setup}(1^\lambda, 1^k) \\ (\mathsf{com}, i, r_i^*, \pi_i) & \leftarrow \widetilde{\mathcal{P}}(\mathsf{crs}) \\ r & \leftarrow \mathsf{Open}(1^k, \mathsf{crs}, \mathsf{com}) \end{array}\right] \le \mathsf{negl}(\lambda).$$

[6]The set of commitments $\mathcal{COM}$ should not be thought of as the set of all valid commitments (and indeed it may contain commitments not in the support of $\mathsf{GenBits}$). In particular, the simplest way to satisfy this property is to bound the bit-length of com and have the verifier reject commitments that are too large. Note that additional structural properties about com can be checked by the Verify algorithm.

Computationally Hiding: *We require that for all polynomial $k = k(\lambda)$ and $I \subseteq [k]$, the two following distributions are computationally indistinguishable:*

$$\Big(\mathsf{crs}, \mathsf{com}, I, r_I, \pi_I, r_{\bar{I}}\Big) \\ \overset{c}{\approx} \\ \Big(\mathsf{crs}, \mathsf{com}, I, r_I, \pi_I, r'_{\bar{I}}\Big),$$

where $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^k)$, $(\mathsf{com}, r, \pi_{[k]}) \leftarrow \mathsf{GenBits}(\mathsf{crs})$ *and* $r' \xleftarrow{\$} \{0,1\}^k$.

Designated-Verifier Hidden-Bits Generator. We define the Designated-Verifier version of a Hidden-Bits Generator (DV-HBG) similarly, but with the following differences:

- $\mathsf{Setup}(1^\lambda, 1^k)$: Now outputs $(\mathsf{crs}, \mathsf{td})$, where td is a trapdoor associated to the crs;
- $\mathsf{Verify}(\mathsf{crs}, \mathsf{td}, \mathsf{com}, i, r_i, \pi_i)$ takes the trapdoor td as an additional input, and outputs `accept` or `reject` as before;
- For *Statistical Binding*, the cheating prover $\widetilde{\mathcal{P}}$ can now make a polynomial number of oracle queries to $\mathsf{Verify}(\mathsf{crs}, \mathsf{td}, \cdots)$. We require that for any such $\widetilde{P}$:

$$\Pr\left[\begin{array}{ll} & r_i^* \neq r_i \\ \wedge & \mathsf{Verify}(\mathsf{crs}, \mathsf{td}, \mathsf{com}, i, r_i^*, \pi_i) = \mathtt{accept} \end{array} : \begin{array}{cl} (\mathsf{crs}, \mathsf{td}) & \leftarrow \mathsf{Setup}(1^\lambda, 1^k) \\ (\mathsf{com}, i, r_i^*, \pi_i) & \leftarrow \widetilde{\mathcal{P}}^{\mathsf{Verify}(\mathsf{crs}, \mathsf{td}, \cdots)}(\mathsf{crs}) \\ r & \leftarrow \mathsf{Open}(1^k, \mathsf{crs}, \mathsf{com}) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

- For *Computational Hiding*, we require that the distributions are indistinguishable given the associated trapdoor td:

$$\Big(\mathsf{crs}, \mathsf{td}, \mathsf{com}, I, r_I, \pi_I, r_{\bar{I}}\Big) \overset{c}{\approx} \Big(\mathsf{crs}, \mathsf{td}, \mathsf{com}, I, r_I, \pi_I, r'_{\bar{I}}\Big),$$

  where $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda, 1^k)$, $(\mathsf{com}, r, \pi_{[k]}) \leftarrow \mathsf{GenBits}(\mathsf{crs})$ and $r' \xleftarrow{\$} \{0,1\}^k$.

4 From Hidden-Bits Generator to NIZKs

We now prove that we can combine any (DV-)HBG with a NIZK in the Hidden-Bits model to get a (Reusable DV-)NIZK in the CRS model. Recall that our basic notion of NIZKs considered selective version of ZK where the statement to be proven is chosen prior to seeing the CRS. In Section 4.2 we will then extend our compiler to the adaptive ZK setting.

Theorem 4.1. *Suppose there exists a Hidden-Bits Generator, then there exists a publicly verifiable NIZK. Suppose there exists a designated-verifier Hidden-Bits Generator (DV-HBG), then there exists a reusable designated-verifier NIZK (reusable DV-NIZK).*

4.1 Proof of Theorem 4.1

For simplicity, we first consider the publicly verifiable version of Theorem 4.1, and then discuss the minor differences that are needed to extend it to the designated-verifier setting.

Construction. Let L be an NP language and n be an integer. Let ($\mathsf{Setup}^{\mathsf{BG}}$, $\mathsf{GenBits}$, Verify) be a hidden-bits generator (Definition 3.1), where $|\mathcal{COM}| = |\mathcal{COM}(\lambda)| \leq 2^{k^\delta p(\lambda)}$ for some polynomial p and constant $\delta < 1$. (where k is the number of hidden bits generated).

Given a NIZK in the hidden-bits model for L using $k' = k'(\lambda, n)$ hidden bits (which exists unconditionally by Theorem 2.8), by Remark 2.7, there exists, for all polynomial $q(\lambda, n)$ (which we will set later), a NIZK in the hidden-bits model $(\mathcal{P}^{\mathsf{HB}}, \mathcal{V}^{\mathsf{HB}})$ using $k = k' \cdot q(\lambda, n)$ hidden bits with soundness-error $2^{-q(\lambda,n)} \cdot \mathsf{negl}(\lambda)$.

Consider the following candidate NIZK ($\mathsf{Setup}^{\mathsf{ZK}}, \mathcal{P}, \mathcal{V}$) in the CRS model:

- $\mathsf{Setup}^{\mathsf{ZK}}(1^\lambda, 1^n)$: Compute $\mathsf{crs}^{\mathsf{BG}} \leftarrow \mathsf{Setup}^{\mathsf{BG}}(1^\lambda, 1^k)$, sample $s \overset{\$}{\leftarrow} \{0,1\}^k$ and output:
$$\mathsf{crs} = (\mathsf{crs}^{\mathsf{BG}}, s);$$
- $\mathcal{P}(\mathsf{crs}, x, w)$: Compute $(\mathsf{com}, r^{\mathsf{BG}}, \pi_{[k]}) \leftarrow \mathsf{GenBits}(\mathsf{crs}^{\mathsf{BG}})$. Set $r_i = r_i^{\mathsf{BG}} \oplus s_i$ for all $i \in [k]$, and run the hidden-bits prover to get $(I \subseteq [k], \pi^{\mathsf{HB}}) \leftarrow \mathcal{P}^{\mathsf{HB}}(r, x, w)$. Output:
$$\Pi = (I, \pi^{\mathsf{HB}}, \mathsf{com}, r_I, \pi_I).$$
- $\mathcal{V}(\mathsf{crs}, x, \Pi = ((I, \pi^{\mathsf{HB}}, \mathsf{com}, r_I, \pi_I)))$: Compute $r_i^{\mathsf{BG}} = r_i \oplus s_i$ for all $i \in [k]$. Accept if for all $i \in I$, $\mathsf{Verify}(\mathsf{crs}^{\mathsf{BG}}, \mathsf{com}, i, r_i^{\mathsf{BG}}, \pi_i)$ accepts, and if $\mathcal{V}^{\mathsf{HB}}(I, r_I, x, \pi^{\mathsf{HB}})$ also accepts.

Completeness. By completeness of the HBG, $\mathsf{Verify}(\mathsf{crs}^{\mathsf{BG}}, \mathsf{com}, i, r_i^{\mathsf{BG}}, \pi_i)$ accepts for all $i \in I$, and by completeness of the hidden-bits NIZK, $\mathcal{V}^{\mathsf{HB}}(I, r_I, x, \pi^{\mathsf{HB}})$ also accepts.

Soundness. Suppose the (unbounded) cheating prover $\widetilde{\mathcal{P}}(\mathsf{crs})$ has a $\mu(\lambda)$ probability of outputting $x \notin L$ and $\Pi^* = (I, \pi^{HB}, \mathsf{com}, r_I^*, \pi_I)$ such that Π^* is an accepting proof:

$$\mu(\lambda) \overset{\text{def}}{=} \Pr\left[\begin{array}{c} \mathcal{V}(\mathsf{crs}, x, \Pi^*) = \mathtt{accept} \\ \wedge \\ x \notin L \end{array} : \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{Setup}^{\mathsf{ZK}}(1^\lambda, 1^n) \\ (x, \Pi^* = (I, \pi^{HB}, \mathsf{com}^*, r_I^*, \pi_I)) \leftarrow \widetilde{\mathcal{P}}(\mathsf{crs}) \end{array}\right]$$

By statistical binding of the HBG, we know that if Π^* is accepting then with all but negligible probability $r_I^* = r_I^{\mathsf{BG}} \oplus s_I$ where $r^{\mathsf{BG}} = \mathsf{Open}(1^k, \mathsf{crs}^{\mathsf{BG}}, \mathsf{com}^*)$. Furthermore, by succinctness of HBG, we know that $\mathsf{com}^* \in \mathcal{COM}(\lambda)$. Therefore, the prover's success probability can only go down negligibly if we replace r_I^* by $r_I = r_I^{\mathsf{BG}} \oplus s_I$ and we require $\mathsf{com}^* \in \mathcal{COM}(\lambda)$. This means:

$$\nu(\lambda) \overset{\text{def}}{=} \Pr\left[\begin{array}{ll} & \mathcal{V}^{\mathsf{HB}}(I, r_I, x, \pi^{\mathsf{HB}}) = \mathtt{accept} \\ \wedge & x \notin L \\ \wedge & \mathsf{com}^* \in \mathcal{COM}(\lambda) \end{array} : \begin{array}{l} \mathsf{crs} = (\mathsf{crs}^{\mathsf{BG}}, s) \leftarrow \mathsf{Setup}^{\mathsf{ZK}}(1^\lambda, 1^n) \\ (x, \Pi^* = (I, \pi^{HB}, \mathsf{com}^*, r_I^*, \pi_I)) \leftarrow \widetilde{\mathcal{P}}(\mathsf{crs}) \\ r^{\mathsf{BG}} = \mathsf{Open}(1^k, \mathsf{crs}^{\mathsf{BG}}, \mathsf{com}^*),\ r = r^{\mathsf{BG}} \oplus s \end{array}\right]$$
$$\geq \mu(\lambda) - \mathsf{negl}(\lambda).$$

Fix any $\mathsf{com} \in \mathcal{COM}(\lambda)$. Define the probability of the prover winning the previous game and satisfying $\mathsf{com}^* = \mathsf{com}$:

$$\begin{aligned}
\nu_{\mathsf{com}}(\lambda) &\overset{\text{def}}{=} \Pr\left[\begin{array}{ll} \quad \mathcal{V}^{\mathsf{HB}}(I, r_I, x, \pi^{\mathsf{HB}}) = \mathtt{accept} & \mathsf{crs} = (\mathsf{crs}^{\mathsf{BG}}, s) \leftarrow \mathsf{Setup}^{\mathsf{ZK}}(1^\lambda, 1^n) \\ \wedge \quad x \notin L & : \quad (x, \Pi^* = (I, \pi^{HB}, \mathsf{com}^*, r_I^*, \pi_I)) \leftarrow \widetilde{\mathcal{P}}(\mathsf{crs}) \\ \wedge \quad \mathsf{com}^* = \mathsf{com} & r^{\mathsf{BG}} = \mathsf{Open}(1^k, \mathsf{crs}^{\mathsf{BG}}, \mathsf{com}^*),\ r = r^{\mathsf{BG}} \oplus s \end{array}\right] \\
&\le \Pr\left[\begin{array}{ll} \mathcal{V}^{\mathsf{HB}}(I, r_I, x, \pi^{\mathsf{HB}}) = \mathtt{accept} & \mathsf{crs} = (\mathsf{crs}^{\mathsf{BG}}, s) \leftarrow \mathsf{Setup}^{\mathsf{ZK}}(1^\lambda, 1^n) \\ \wedge & : \quad r^{\mathsf{BG}} = \mathsf{Open}(1^k, \mathsf{crs}^{\mathsf{BG}}, \mathsf{com}), r = r^{\mathsf{BG}} \oplus s \\ x \notin L & (x, I, \pi^{HB}) \leftarrow \widehat{P}_{\mathsf{crs}^{\mathsf{BG}},\mathsf{com}}(r) \end{array}\right] \\
&\le 2^{-q(\lambda,n)} \cdot \mathsf{negl}(\lambda).
\end{aligned}$$

In the first inequality above, we define $\widehat{P}_{\mathsf{crs}^{\mathsf{BG}},\mathsf{com}}(r)$ to run $\widetilde{\mathcal{P}}(\mathsf{crs}^{\mathsf{BG}}, \mathsf{Open}(1^k, \mathsf{crs}^{\mathsf{BG}}, \mathsf{com}) \oplus r)$ and output the relevant components. The second inequality then follows from soundness of the NIZK in the hidden-bits model by thinking of $\widehat{P}_{\mathsf{crs}^{\mathsf{BG}},\mathsf{com}}$ as a malicious hidden-bits model prover and noting that $r = r^{\mathsf{BG}} \oplus s$ is uniformly random over the choice of s.

Setting $q(\lambda, n) = (k'(\lambda, n) \cdot p(\lambda))^{1/(1-\delta)}$, we have:

$$k^\delta \cdot p(\lambda) = (k' \cdot q(\lambda, n))^\delta \cdot p(\lambda) \le q^\delta(\lambda, n) \cdot q^{1-\delta}(\lambda, n),$$

so that $|\mathcal{COM}| \cdot \nu_{\mathsf{com}}(\lambda) \le \mathsf{negl}(\lambda)$. Combining the above, and using an union bound we get:

$$\mu(\lambda) - \mathsf{negl}(\lambda) \le \nu(\lambda) \le \sum_{\mathsf{com} \in \mathcal{COM}(\lambda)} \nu_{\mathsf{com}}(\lambda) \le \mathsf{negl}(\lambda)$$

which shows that $\mu(\lambda) = \mathsf{negl}(\lambda)$ and concludes the proof of soundness.

Zero-Knowledge. Let $\mathsf{Sim}^{\mathsf{HB}}$ be a hidden-bits simulator (given by zero-knowledge). Define the following simulator for the NIZK:

- $\mathsf{Sim}(1^\lambda, x)$:
  Compute $(I, r_I, \pi^{\mathsf{HB}}) \leftarrow \mathsf{Sim}^{\mathsf{HB}}(x)$.
  Compute $\mathsf{crs}^{\mathsf{BG}} \leftarrow \mathsf{Setup}^{\mathsf{BG}}(1^\lambda, 1^k)$, and $(\mathsf{com}, r^{\mathsf{BG}}, \pi_{[k]}) \leftarrow \mathsf{GenBits}(\mathsf{crs}^{\mathsf{BG}})$.
  Set for all $i \in [k]$: $s_i = r_i \oplus r_i^{\mathsf{BG}}$ if $i \in I$; and $s_i \overset{\$}{\leftarrow} \{0,1\}$ otherwise.
  Output $\mathsf{crs} = (\mathsf{crs}^{\mathsf{BG}}, s_{[k]})$, $\Pi = (I, \pi^{\mathsf{HB}}, \mathsf{com}, r_I, \pi_I)$.

We now prove that no PPT adversary $\mathcal{A}$ can distinguish between (crs, Π) generated by the Real or the Ideal experiment (see Definition 2.2). The proof proceeds via a sequence of hybrids.

$\mathbf{H_0}$: This is the Real experiment. The adversary chooses $(x, w) \in R_L$ with $|x| = n$. Let $k = k(\lambda, n)$. The experiment proceeds as follows:

- $\mathsf{crs}^{\mathsf{BG}} \leftarrow \mathsf{Setup}^{\mathsf{BG}}(1^\lambda, 1^k), s \overset{\$}{\leftarrow} \{0,1\}^k$
- $(\mathsf{com}, r^{\mathsf{BG}}, \pi_{[k]}) \leftarrow \mathsf{GenBits}(\mathsf{crs}^{\mathsf{BG}})$

- $r_i := r_i^{\mathsf{BG}} \oplus s_i$ for all $i \in [k]$
- $(I, \pi^{\mathsf{HB}}) \leftarrow \mathcal{P}^{\mathsf{HB}}(r, x, w)$

Output $\mathsf{crs} = (\mathsf{crs}^{\mathsf{BG}}, s), \Pi = (I, \pi^{\mathsf{HB}}, \mathsf{com}, r_I, \pi_I)$.

$\mathbf{H_1}$: In this experiment we switch how r, s are picked: we first pick $r \xleftarrow{\$} \{0,1\}^k$, and then set $s = r \oplus r^{\mathsf{BG}}$. This allows us to reorder the operations as follows:

- $r \xleftarrow{\$} \{0,1\}^k$
- $(I, \pi^{\mathsf{HB}}) \leftarrow \mathcal{P}^{\mathsf{HB}}(r, x, w)$
- $\mathsf{crs}^{\mathsf{BG}} \leftarrow \mathsf{Setup}^{\mathsf{BG}}(1^\lambda, 1^k)$, $(\mathsf{com}, r^{\mathsf{BG}}, \pi_{[k]}) \leftarrow \mathsf{GenBits}(\mathsf{crs}^{\mathsf{BG}})$
- $s_i := r_i^{\mathsf{BG}} \oplus r_i$ for all $i \in [k]$

Output $\mathsf{crs} = (\mathsf{crs}^{\mathsf{BG}}, s), \Pi = (I, \pi^{\mathsf{HB}}, \mathsf{com}, r_I, \pi_I)$.

The hybrid H_1 is distributed identically to H_0.

$\mathbf{H_2}$: We now switch how s is computed.

- $r \xleftarrow{\$} \{0,1\}^k$
- $(I, \pi^{\mathsf{HB}}) \leftarrow \mathcal{P}^{\mathsf{HB}}(r, x, w)$
- $\mathsf{crs}^{\mathsf{BG}} \leftarrow \mathsf{Setup}^{\mathsf{BG}}(1^\lambda, 1^k)$, $(\mathsf{com}, r^{\mathsf{BG}}, \pi_{[k]}) \leftarrow \mathsf{GenBits}(\mathsf{crs}^{\mathsf{BG}})$
- $s_i = r_i \oplus r_i^{\mathsf{BG}}$ if $i \in I$ and $s_i \xleftarrow{\$} \{0,1\}$ otherwise.

Output $\mathsf{crs} = (\mathsf{crs}^{\mathsf{BG}}, s), \Pi = (I, \pi^{\mathsf{HB}}, \mathsf{com}, r_I, \pi_I)$.

The view of the adversary in H_2 is indistinguishable from H_1 by the computationally hiding property of the HBG. In particular, this property ensures that, even given all the other components seen by the adversary in Hybrid H_1 the values r_i^{BG} are indistinguishable from uniform for $i \notin I$. Note that the set I is chosen a-priori before generating any of the components of the HBG.

$\mathbf{H_3}$: We now switch how $(I, r_I, \pi^{\mathsf{HB}})$ are computed by using $\mathsf{Sim}^{\mathsf{HB}}(x)$ to generate them:

- $(I, r_I, \pi^{\mathsf{HB}}) \leftarrow \mathsf{Sim}^{\mathsf{HB}}(x)$
- $\mathsf{crs}^{\mathsf{BG}} \leftarrow \mathsf{Setup}^{\mathsf{BG}}(1^\lambda, 1^k)$, $(\mathsf{com}, r^{\mathsf{BG}}, \pi_{[k]}) \leftarrow \mathsf{GenBits}(\mathsf{crs}^{\mathsf{BG}})$
- $s_i = r_i \oplus r_i^{\mathsf{BG}}$ if $i \in I$ and $s_i \xleftarrow{\$} \{0,1\}$ otherwise.

Output $\mathsf{crs} = (\mathsf{crs}^{\mathsf{BG}}, s), \Pi = (I, \pi^{\mathsf{HB}}, \mathsf{com}, r_I, \pi_I)$.

Hybrid H_3 is statistically indistinguishable from H_2 by the zero-knowledge property of the hidden-bits NIZK.

Note that hybrid H_3 is equivalent to the Ideal experiment with our simulator. Therefore the above hybrids show that the Real and Ideal experiments are computationally indistinguishable as needed to prove the zero-knowledge property.

Reusable DV-NIZK from DV-HBG: The above construction of NIZKs from a Hidden-Bits Generator (HBG) and its proof of security extend naturally to the designated-verifier setting to get a construction of reusable DV-NIZKs from DV-HBG. This is essentially immediate and we outline the minor syntactic changes that are needed in the designated-verifier setting.

In the construction, the NIZK setup $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Setup}^{\mathsf{ZK}}(1^\lambda, 1^n)$ now also generates a trapdoor td which it gets from $\mathsf{Setup}^{\mathsf{BG}}$ of the DV-HBG. The NIZK verification algorithm $\mathcal{V}(\mathsf{crs}, \mathsf{td}, x, \Pi)$ now needs to use this trapdoor to check the HBG proofs by running $\mathsf{Verify}(\mathsf{crs}^{\mathsf{BG}}, \mathsf{td}, \mathsf{com}, i, r_i^{\mathsf{BG}}, \pi_i)$.

In the proof of soundness, the only difference is that the cheating prover $\widetilde{P}$ now can make polynomially many queries to the NIZK verification algorithm $\mathcal{V}(\mathsf{crs}, \mathsf{td}, \cdots)$ with the trapdoor td which translates to making polynomially many queries to the DV-HBG verification algorithm $\mathsf{Verify}(\mathsf{crs}^{\mathsf{BG}}, \mathsf{td}, \cdots)$. The proof remains identical otherwise, and when we use the statistical soundness of the DV-HBG we rely on the fact that it holds even given polynomially queries to the DV-HBG verification algorithm $\mathsf{Verify}(\mathsf{crs}^{\mathsf{BG}}, \mathsf{td}, \cdots)$.

In the proof of zero-knowledge, the only difference is that we now sample $(\mathsf{crs}^{\mathsf{BG}}, \mathsf{td}) \leftarrow \mathsf{Setup}^{\mathsf{BG}}(1^\lambda, 1^k)$ and include the trapdoor td in the view of the adversary. The entire sequence of hybirds is otherwise identical and when we use the computational hiding property of the HBG, we rely on the fact that it holds even given td.

4.2 Adaptive ZK

Our default definition of (reusable designated-verifier) NIZKs considers a *selective* version of the zero-knowledge property, where the statement x is chosen before the CRS. We now also consider a stronger *adaptive* zero-knowledge property, where the statement x can depend adaptively on the CRS. Let us begin by defining adaptive ZK.

Definition 4.2 (Adaptive ZK)**.** *A (reusable designated-verifier) NIZK satisfies adaptive Zero-Knowledge (adaptive ZK) if the following holds. We require that there exists a stateful PPT simulator* Sim *such that for any stateful PPT adversary* $\mathcal{A}$ *the two following distributions are computationally indistinguishable:*

```
EXP^{Real}(1^λ) :

$1^n \leftarrow \mathcal{A}(1^\lambda)$
$(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$
$(x, w) \leftarrow \mathcal{A}(\mathsf{crs}, \mathsf{td})$
        where $(x, w) \in R_L, |x| = n$
$\pi \leftarrow \mathcal{P}(\mathsf{crs}, x, w)$
Output $\mathcal{A}(\pi)$
```

```
EXP^{Ideal}(1^λ) :

$1^n \leftarrow \mathcal{A}(1^\lambda)$
$(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{Sim}(1^\lambda, 1^n)$
$(x, w) \leftarrow \mathcal{A}(\mathsf{crs}, \mathsf{td})$
        where $(x, w) \in R_L, |x| = n$
$\pi \leftarrow \mathsf{Sim}(x)$
Output $\mathcal{A}(\pi)$
```

Outline of Our Results. We show how to extend Theorem 4.1 to the adaptive setting but in doing so we need to overcome some fairly subtle issues.

Initially, one may hope that the same proof of selective ZK security would naturally extend to the adaptive setting. However, this appears to already fail when we try to define an adaptive ZK simulator. Recall that in the selective setting, the simulator first ran the hidden-bits simulator $(I, r_I, \pi^{\mathsf{HB}}) \leftarrow \mathsf{Sim}^{\mathsf{HB}}(x)$ and then carefully "programmed" the CRS by choosing s to ensure that the values of the opened hidden bits was r_I. But that means that the simulator needed to know the statement x before outputting the CRS while in the adaptive setting it would need to create the CRS before knowing x. To overcome this, we need to rely on special properties of the NIZK in the hidden-bits model.

As a start, we identified the following property (*) of NIZKs in the hidden-bits model: we can break up the simulator into two components: $r \leftarrow \mathsf{Sim}_1^{\mathsf{HB}}$ chooses some value r, which may not be uniformly random, and $(I, \pi^{\mathsf{HB}}) \leftarrow \mathsf{Sim}_2^{\mathsf{HB}}(x)$ outputs some values depending on x such that the resultant distribution of $(I, r_I, \pi^{\mathsf{HB}})$ is statistically close to the output of the real prover. Using property (*), we can already define a meaningful ZK simulator for the compiled NIZK. In particular, the simulator would program the CRS by choosing s to ensure that the values of the hidden bits was $r \leftarrow \mathsf{Sim}_1^{\mathsf{HB}}$. However, we were unable to prove indistinguishability with this simulator. Indeed, the difficulty is related to subtle "selective-opening" issues: the fact that the adversary can choose the statement x and therefore also influence the choice I of positions to open adaptively after seeing the CRS appears to prevent natural reductions. We leave it open whether one can generically show that the compiler achieves adaptive ZK security if one starts from an arbitrary hidden-bits NIZK which is guaranteed to satisfying property (*); we conjecture that the answer is negative and that one can get "unnatural counterexamples" along the lines of [HRW16].

Instead we identify an even stronger property of the NIZK in the hidden bits model which we refer to as "special ZK" and which does suffice to get the proof to go through. For special ZK we require that there is a two-part simulator $(\mathsf{Sim}_1^{\mathsf{HB}}, \mathsf{Sim}_2^{\mathsf{HB}})$ with the syntax $r' = \mathsf{Sim}_1^{\mathsf{HB}}(r)$ and $(I, \pi) = \mathsf{Sim}_2^{\mathsf{HB}}(x, r)$. We want to ensure that if $r' = \mathsf{Sim}_1^{\mathsf{HB}}(r)$ and if the honest hidden-bits prover outputs $(I, \pi) \leftarrow \mathcal{P}(r, x, w)$ it must be the case that $r_I = r'_I$. In other words, the honest prover only opens bits on which r and r' agree. Furthermore if r is random and $r' = \mathsf{Sim}_1^{\mathsf{HB}}(r)$ then the distribution of (r', I, π) is statistically close whether $(I, \pi) \leftarrow \mathcal{P}(r, x, w)$ or $(I, \pi) \leftarrow \mathsf{Sim}_2^{\mathsf{HB}}(x, r)$. With the above special ZK property, we are able to show that the same compiler from the previous section also achieves adaptive ZK security. In particular, we can rely on the computational hiding property of the HBG to switch between programming the CRS to r and r' while knowing that the values of r, r' on the set I of positions that will be opened are the same in both cases since $r_I = r'_I$. In other words, the set of positions that needs to be switched when going from a real CRS to a simulated CRS is $\{i : r_i \neq r'_i\}$ which is fixed and does not itself depend adaptively on the CRS.

Special-ZK in Hidden-Bits Model. We define a restricted notion of ZK in the hidden-bits model, which we refer to as *special-ZK*. Jumping ahead, we will use NIZKs in the hidden-bits model with the special-ZK property in order to get publicly verifiable or designated-verifier NIZKs with adaptive ZK.

Definition 4.3 (Special ZK)**.** *A NIZK $(\mathcal{P}, \mathcal{V})$ in the Hidden Bits Model satisfies the special-ZK property if there exists an efficient simulator $\mathsf{Sim}_1, \mathsf{Sim}_2$ such that the following properties hold.*

1. *For any $(x, w) \in R_L$ and any $r \in \{0,1\}^{k(\lambda,|x|)}$ if we let $r' = \mathsf{Sim}_1(r)$ and $(I, \pi) \leftarrow \mathcal{P}(1^\lambda, r, x, w)$ then $r'_I = r_I$.*
2. *For any adversary $\mathcal{A}$ and any $n = n(\lambda)$, the two following distributions are statistically indistinguishable:*
$$(I, r', \pi) \overset{s}{\approx} (I', r', \pi')$$
where $r \leftarrow \{0,1\}^{k(\lambda,n)}, r' = \mathsf{Sim}_1(r), (x, w) \leftarrow \mathcal{A}(r'), (I, \pi) \leftarrow \mathcal{P}(1^\lambda, r, x, w)$, $(I', \pi') \leftarrow \mathsf{Sim}_2(r, x)$.

Note that special-ZK implies the original ZK property in the hidden bits model. This is because: by property (1) the honestly generated proof (I, r_I, π) is identical to (I, r'_I, π) where $r' = \mathsf{Sim}_1(r)$, and by property (2) we have that (I, r'_I, π) is statistically close to (I', r'_I, π').

Lemma 4.4. *For every $L \in NP$, there exists a NIZK in the hidden-bits model satisfying the special-ZK property (see Definition 4.3).*

Proof Sketch. We show that the classical NIZK construction of Theorem 2.8 (due to [FLS99]) satisfies the special ZK property. For ease of presentation, we first give the construction in a variant of the hidden-bits model where the bits r can come from some designated distribution and then discuss how to extend it to the case where r is uniformly ranodm.

The protocol of [FLS99] is for the NP complete language of Hamiltonicity and proceeds as follows. Consider an input graph $G = (V, E)$ on $n = |V|$ vertices. The hidden-bits r specify a random cycle graph C on n vertices, represented by its $n \times n$ adjacency matrix. Given as input a Hamiltonian graph G (together with a corresponding Hamiltonian cycle), the prover finds an injective mapping $\pi : V \to C$ that preserves the cycle structure. The prover sends the mapping π and also reveals all the entries in C that correspond to *non-edges* of $\pi(G) = \{(\pi(u), \pi(v)) : (u, v) \in E\}$.

Given as input G, the proof π and the revealed entries of C, the verifier checks that (1) π is a permutation and (2) that every non-edge of G was indeed mapped to a revealed non-edge of C (i.e., a 0 in the adjacency matrix).

The fact that this protocol is an NIZK in the hidden bits model is established in [FLS99] and here we focus on showing it satisfies the special ZK property (as in Definition 4.3).

Consider a simulator Sim_1 that simply outputs the all 0's string. Since the honest prover only ever reveals 0 entries (i.e., non-edges), the first condition of special ZK holds trivially.

To see that the second condition holds, consider a simulator Sim_2, that given as input the (Hamiltonian) graph $G = (V, E)$ chooses a random permutation π' of the vertices, constructs a graph $C' = \{(\pi(u), \pi(v)) : (u, v) \in E\}$ and outputs (I', π') where $I' \subseteq V$ corresponds to the set of non-edges in C'.

We first argue that π and π' are identically distributed. This follows from the fact that π is an (injective) mapping to a *random* cycle and is therefore a random injective function. Next, we observe that I and I' are computed by applying the same function to (G, π) and (G, π'), respectively (specifically, the function that generates the graph induced by π and outputs all the non-edges). Therefore (I, π) and (I', π') are identically distributed, as required.

Handling Uniform Hidden Bits. Following the presentation in [Gol01, Section 4.10.2], we first describe a construction that uses uniformly distributed random bits, but only achieves an inverse polynomial gap between completeness and soundness.

The idea is for the hidden bits to be an $n^5 \times n^5$ uniformly distributed matrix M. It is shown in [FLS99, Gol01] that with some inverse polynomial probability this matrix contains as a generalized sub-matrix, the adjacency matrix of a cycle graph (and this property can be efficiently checked). We say that such a matrix M is *useful*.

In case M is not useful, the prover reveals all of the hidden bits and the verifier (after checking that M is indeed useful) immediately accepts. Otherwise (i.e., if M is useful) the prover and verifier run the procedure described above using the cycle graph embedded within M as the hidden bit string. Perfect completeness is immediate and the fact that the soundness error is bounded by $1 - 1/\mathsf{poly}(n)$ follows from the above discussion.

For special zero-knowledge (as in Definition 4.3), we construct a simulator $(\mathsf{Sim}_1, \mathsf{Sim}_2)$ as follows. Recall that the simulator Sim_1 is given as input a random string M specifying a hidden bit string. In case M is not useful, Sim_1 simply outputs M and otherwise Sim_1 outputs the all-zeros string.

As for Sim_2, in case M is not useful, it outputs I' specifying the entire matrix M and an empty proof π' (since in the corresponding real interaction there is no proof and the verifier immediately accepts). In case M is useful, Sim_2 operates as discussed in the setting of the non-uniform hidden bit string.

The first condition of special ZK holds immediately (since in the "not useful case" the entire hidden bit string is opened and in the "useful case" only 0's are revealed). The second condition holds trivially in the not useful case and by the above discussion also in the useful case.

To obtain a construction with negligible soundness error, the above protocol is repeated $\mathsf{poly}(n)$ times (in parallel), while observing that parallel repetition reduces the soundness error at an exponential rate and preserves the special zero-knowledge property (since we can run the two simulators independently $\mathsf{poly}(n)$ times). $\square$

The Adaptive Compiler. We are now ready to extend our compiler from Theorem 4.1 to the adaptive setting.

Theorem 4.5. *Suppose there exists a Hidden-Bits Generator, then there exists a publicly verifiable NIZK with adaptive ZK security. Suppose there exists a designated-verifier*

Hidden-Bits Generator (DV-HBG), then there exists a reusable designated-verifier NIZK (DV-NIZK) with adaptive ZK security.

Proof. We use the same construction as in the selective case, but require the underlying NIZK in the hidden-bits model to satisfy the stronger special-ZK property. The proof of completeness and soundness is therefore identical to the selective case and we are only left to show adaptive ZK. For simplicity, we begin by giving the proof in the publicly-verifiable setting, but the proof extends naturally to the designated-verifier setting by just adding the HBG trapdoor to all the views.

Let $\mathsf{Sim}_1^{\mathsf{HB}}, \mathsf{Sim}_2^{\mathsf{HB}}$ be a special-ZK simulator of the underlying hidden-bits NIZK. We define the following simulator for the constructed NIZK:

- Sim:
  - On input $(1^\lambda, 1^n)$: Let $k = k(\lambda, n)$
    Sample $\mathsf{crs}^{\mathsf{BG}} \leftarrow \mathsf{Setup}^{\mathsf{BG}}(1^\lambda, 1^k)$, $r \leftarrow \{0,1\}^k$. Let $r' = \mathsf{Sim}_1^{\mathsf{HB}}(r)$.
    Sample $(\mathsf{com}, r^{\mathsf{BG}}, \pi_{[k]}) \leftarrow \mathsf{GenBits}(\mathsf{crs}^{\mathsf{BG}})$. Set $s = r^{\mathsf{BG}} \oplus r'$.
    Output $\mathsf{crs} = (\mathsf{crs}^{\mathsf{BG}}, s)$.
  - On input x:
    Compute $(I, \pi^{\mathsf{HB}}) \leftarrow \mathsf{Sim}_2^{\mathsf{HB}}(r, x)$.
    Output $\Pi = (I, \pi^{\mathsf{HB}}, \mathsf{com}, r_I, \pi_I)$,

We now prove that no PPT adversary $\mathcal{A}$ can distinguish between (crs, Π) generated by the Real or the Ideal experiment of Definition 4.2. The proof proceeds via a sequence of hybrids.

$\mathbf{H_0}$: This is the Real experiment:

- The adversary specifies $(1^\lambda, 1^n)$. Let $k = k(\lambda, n)$.
  - $\mathsf{crs}^{\mathsf{BG}} \leftarrow \mathsf{Setup}^{\mathsf{BG}}(1^\lambda, 1^k)$, $s \overset{\$}{\leftarrow} \{0,1\}^k$
  - Output $\mathsf{crs} = (\mathsf{crs}^{\mathsf{BG}}, s)$.
- The adversary specifies $(x, w) \in R_L$.
  - $(\mathsf{com}, r^{\mathsf{BG}}, \pi_{[k]}) \leftarrow \mathsf{GenBits}(\mathsf{crs}^{\mathsf{BG}})$
  - $r_i := r_i^{\mathsf{BG}} \oplus s_i$ for all $i \in [k]$
  - $(I, \pi^{\mathsf{HB}}) \leftarrow \mathcal{P}^{\mathsf{HB}}(r, x, w)$
  - Output $\Pi = (I, \pi^{\mathsf{HB}}, \mathsf{com}, r_I, \pi_I)$.

$\mathbf{H_1}$: In this experiment we switch how r, s are picked: we first pick $r \overset{\$}{\leftarrow} \{0,1\}^k$, and then set $s = r \oplus r^{\mathsf{BG}}$. We reorder the operations as follows:

- The adversary specifies $(1^\lambda, 1^n)$. Let $k = k(\lambda, n)$.
  - $r \overset{\$}{\leftarrow} \{0,1\}^k$

  - $\mathsf{crs}^{\mathsf{BG}} \leftarrow \mathsf{Setup}^{\mathsf{BG}}(1^\lambda, 1^k)$
  - $(\mathsf{com}, r^{\mathsf{BG}}, \pi_{[k]}) \leftarrow \mathsf{GenBits}(\mathsf{crs}^{\mathsf{BG}})$
  - $s_i := r_i^{\mathsf{BG}} \oplus r_i$ for all $i \in [k]$
  - Output $\mathsf{crs} = (\mathsf{crs}^{\mathsf{BG}}, s)$.

- The adversary specifies $(x, w) \in R_L$.
  - $(I, \pi^{\mathsf{HB}}) \leftarrow \mathcal{P}^{\mathsf{HB}}(r, x, w)$
  - Output $\Pi = (I, \pi^{\mathsf{HB}}, \mathsf{com}, r_I, \pi_I)$.

The hybrid H_1 is distributed identically to H_0.

$\mathbf{H_2}$: We now switch how s is computed.

- The adversary specifies $(1^\lambda, 1^n)$. Let $k = k(\lambda, n)$.
  - $r \xleftarrow{\$} \{0,1\}^k$
  - $r' = \mathsf{Sim}_1^{\mathsf{HB}}(r)$
  - $\mathsf{crs}^{\mathsf{BG}} \leftarrow \mathsf{Setup}^{\mathsf{BG}}(1^\lambda, 1^k)$
  - $(\mathsf{com}, r^{\mathsf{BG}}, \pi_{[k]}) \leftarrow \mathsf{GenBits}(\mathsf{crs}^{\mathsf{BG}})$
  - $s_i := r_i^{\mathsf{BG}} \oplus r'_i$ for all $i \in [k]$
  - Output $\mathsf{crs} = (\mathsf{crs}^{\mathsf{BG}}, s)$.

- The adversary specifies $(x, w) \in R_L$.
  - $(I, \pi^{\mathsf{HB}}) \leftarrow \mathcal{P}^{\mathsf{HB}}(r, x, w)$
  - Output $\Pi = (I, \pi^{\mathsf{HB}}, \mathsf{com}, r_I, \pi_I)$.

The view of the Adversary in H_2 is indistinguishable from H_1 by the computationally hiding property of the HBG and condition (1) of the special-ZK property of the hidden-bits NIZK.

Fix any r, r' chosen in the first two steps of the experiment and let $I^* = \{i : r_i = r'_i\}$. We are only changing the distribution of s_i for $i \notin I^*$ between the two hybrids. By the computational hiding property of the HBG, the adversary cannot distinguish the values r_i^{BG} from uniform for $i \notin I^*$ even given $\mathsf{crs}^{\mathsf{BG}}, \mathsf{com}, r_{I^*}, \pi_{I^*}$. Therefore, he cannot distinguish between $s_i := r_i^{\mathsf{BG}} \oplus r_i$ and $s_i := r_i^{\mathsf{BG}} \oplus r'_i$ for $i \notin I^*$. Note that the set I^* is chosen a-priori before generating any of the components of the HBG. Furthermore, by condition (1) of the special-ZK property, we know that the set I chosen later in the experiment has to satisfy $I \subseteq I^*$ and therefore, no matter what (x, w) the adversary chooses, the values r_I, π_I that the adversary gets later are just a subset of r_{I^*}, π_{I^*}.

$\mathbf{H_3}$: We now switch how (I, π^{HB}) is computed.

- The adversary specifies $(1^\lambda, 1^n)$. Let $k = k(\lambda, n)$.
  - $r \xleftarrow{\$} \{0,1\}^k$
  - $r' = \mathsf{Sim}_1^{\mathsf{HB}}(r)$
  - $\mathsf{crs}^{\mathsf{BG}} \leftarrow \mathsf{Setup}^{\mathsf{BG}}(1^\lambda, 1^k)$
  - $(\mathsf{com}, r^{\mathsf{BG}}, \pi_{[k]}) \leftarrow \mathsf{GenBits}(\mathsf{crs}^{\mathsf{BG}})$
  - $s_i := r_i^{\mathsf{BG}} \oplus r'_i$ for all $i \in [k]$
  - Output $\mathsf{crs} = (\mathsf{crs}^{\mathsf{BG}}, s)$.
- The adversary specifies $(x, w) \in R_L$.
  - $(I, \pi^{\mathsf{HB}}) \leftarrow \mathsf{Sim}_2^{\mathsf{HB}}(r, x)$
  - Output $\Pi = (I, \pi^{\mathsf{HB}}, \mathsf{com}, r_I, \pi_I)$.

Hybrid H_3 is statistically indistinguishable from H_2 by condition (2) of the special-ZK property of the Hidden-Bits NIZK. In particular, even if the adversary can choose (x, w) adaptively depending on r' he cannot distinguish between $(r', \mathcal{P}^{\mathsf{HB}}(r, x, w))$ and $(r', \mathsf{Sim}_2^{\mathsf{HB}}(r, x))$.

Note that hybrid H_3 is equivalent to the Ideal experiment with our simulator. Therefore the above hybrids show that the Real and Ideal experiments are computationally indistinguishable, which concludes the proof of adaptive zero-knowledge in the publicly verifiable setting.

To extend the above proof to the designated verifier setting, the only difference is that we now sample $(\mathsf{crs}^{\mathsf{BG}}, \mathsf{td}) \leftarrow \mathsf{Setup}^{\mathsf{BG}}(1^\lambda, 1^k)$ and give the trapdoor td to the adversary together with the crs. The entire sequence of hybrids is otherwise identical and when we use the computational hiding property of the HBG, we rely on the fact that it holds even given td. $\square$

5 Designated-Verifier Hidden-Bits Generator from CDH

Let $(\mathbb{G}, p, g) \leftarrow \mathsf{GroupGen}(1^\lambda)$ be a prime-order group generator so that $\mathbb{G}$ is a group of prime order p, with a generator g. Let hc be the corresponding Goldreich-Levin [GL89] hard-core bit. Let us define the following hidden-bits generator:

- $\mathsf{Setup}(1^\lambda, 1^k)$: Let $(\mathbb{G}, p, g) \leftarrow \mathsf{GroupGen}(1^\lambda)$. For all $i \in [k]$, pick random $a_i, b_i \xleftarrow{\$} \mathbb{Z}_p$ and $h_i \xleftarrow{\$} \mathbb{G}$ and compute:
$$f_i = h_i^{a_i} \cdot g^{b_i}.$$
Sample some random coins γ matching the randomness used by $\mathsf{hc}(\cdot)$. Output:
$$\Big(\mathsf{crs} = \big(\mathbb{G}, \{(h_i, f_i)\}_{i\in[k]}, \gamma\big),\ \mathsf{td} = \{(a_i, b_i)\}_{i\in[k]}\Big).$$

- $\mathsf{GenBits}(\mathsf{crs})$: Pick a random $y \leftarrow \mathbb{Z}_p$, and compute for all $i \in [k]$: $t_i = h_i^y$ and $u_i = f_i^y$. Output:

$$\begin{aligned}\mathsf{com} &= s = g^y, \\ \{r_i &= \mathsf{hc}(t_i; \gamma)\}_{i \in [k]}, \\ \{\pi_i &= (u_i,\, t_i)\}_{i \in [k]}.\end{aligned}$$

- $\mathsf{Verify}(\mathsf{crs}, \mathsf{td} = \{(a_i, b_i)\}, \mathsf{com} = s, i, r_i, \pi_i = (u_i, t_i))$: Compute:

$$\rho_i = t_i^{a_i} \cdot s^{b_i},$$

and accept if and only if $\rho_i = u_i$, and $r_i = \mathsf{hc}(t_i; \gamma)$.

Theorem 5.1. *The triple* $(\mathsf{Setup}, \mathsf{GenBits}, \mathsf{Verify})$ *is a Designated-Verifier Hidden-Bits Generator under CDH.*

Proof. We prove completeness, succinctness, statistical binding and computational hiding.

Completeness: For all $i \in [k]$ we have by construction that $r_i = \mathsf{hc}(t_i; \gamma)$, and for all $h_i \in \mathbb{G}$, $a_i, b_i, y \in \mathbb{Z}_p$, we have:

$$\rho_i = t_i^{a_i} \cdot s^{b_i} = h_i^{y \cdot a_i} \cdot g^{y \cdot b_i} = f_i^y = u_i.$$

Succinctness: We have $|\mathcal{COM}| = |\mathbb{G}| = p$ where $p = 2^{\mathsf{poly}(\lambda)}$ is independent of k. Furthermore since we can test group membership in polynomial time, the verifier rejects if $\mathsf{com} \notin \mathcal{COM} = \mathbb{G}$.

Statistical Binding: The statistical binding property relies on the following lemma.

Lemma 5.2. *[CS98, CS02] Let $\mathbb{G}$ be a group of prime order p with a generator g, and let $h \in \mathbb{G}$. Then for all $(s = g^y, t) \in \mathbb{G}^2$ such that $t \neq h^y$, we have that for $a, b \xleftarrow{\$} \mathbb{Z}_p$:*

$$\left(t^a \cdot s^b,\ h^a \cdot g^b\right) \equiv \mathcal{U}(\mathbb{G} \times \mathbb{G}),$$

over the randomness of $a, b \xleftarrow{\$} \mathbb{Z}_p$, and where $\mathcal{U}(\mathbb{G} \times \mathbb{G})$ is uniform over $\mathbb{G} \times \mathbb{G}$.

Proof of Lemma. Let $x, z \in \mathbb{Z}_p$ be such that $h = g^x, t = g^z$. Then $\left(t^a \cdot s^b,\ h^a \cdot g^b\right) = (g^{az+by}, g^{ax+b})$. Since the exponents are two linearly independent equations in (a, b) they are pairwise independent. □

To prove statistical binding, define $\mathsf{Open}(1^k, \mathsf{crs}, \mathsf{com} = s)$ to inefficiently find y such that $g^y = s$ and output:

$$\{r_i = \mathsf{hc}(h_i^y; \gamma)\}_{i \in [k]}\,.$$

We now argue that for every (potentially unbounded) cheating prover $\widetilde{\mathcal{P}}$:

$$\Pr\left[\begin{array}{ll} & r_i^* \neq r_i \\ \wedge & \mathsf{Verify}(\mathsf{crs}, \mathsf{td}, \mathsf{com}, i, r_i^*, \pi_i) = \mathtt{accept} \end{array} : \begin{array}{cl} (\mathsf{crs}, \mathsf{td}) & \leftarrow \mathsf{Setup}(1^\lambda, 1^k) \\ (\mathsf{com}, i, r_i^*, \pi_i) & \leftarrow \widetilde{\mathcal{P}}^{\mathsf{Verify}(\mathsf{crs}, \mathsf{td}, \cdots)}(\mathsf{crs}) \\ r & \leftarrow \mathsf{Open}(1^k, \mathsf{crs}, \mathsf{com}) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Consider oracle calls from $\widetilde{\mathcal{P}}$ to $\mathsf{Verify}(\mathsf{crs}, \mathsf{td}, \cdots)$, which are of the form $(\mathsf{com} = s, i, r_i, \pi = (t, u))$. Let $y \in \mathbb{Z}_p$ be such that $g^y = s$.

Note also that without loss of generality, $\widetilde{\mathcal{P}}$ does not make any such queries where $t = h_i^y$. This is because the prover can test himself whether any such query accepts or rejects without knowing td. In particular, instead of performing the check $u \overset{?}{=} t^{a_i} \cdot s^{b_i}$ that the verifier would do, the prover can perform the equivalent check $u \overset{?}{=} f_i^y$. Note that since the prover is inefficient he can compute y himself.

Also, without loss of generality, we assume that $\widetilde{\mathcal{P}}$ queries the Verify oracle on his output. Define the event $\mathsf{BadCall}$ to occur if the prover makes a query where $t \neq h_i^y$ and the Verify oracle accepts. We argue that if $\widetilde{\mathcal{P}}$ wins then he must trigger the $\mathsf{BadCall}$ event – this is because, whenever $\widetilde{\mathcal{P}}$ wins, the query corresponding to his output must satisfy $t \neq h_i^y$ and must cause the Verify oracle to accept. Therefore, to prove binding, it suffices to show that the probability of $\mathsf{BadCall}$ occurring is negligible. Let $\mathsf{BadCall}_j$ be the event that the jth query from $\widetilde{\mathcal{P}}$ to Verify is the first such oracle query that causes $\mathsf{BadCall}$ to occur. We argue that for each j, the probability of $\mathsf{BadCall}_j$ occurring is negligible and therefore by union bound the probability of $\mathsf{BadCall}$ is negligible as well. Recall that the Verify oracle only accepts if $t^{a_i} \cdot s^{b_i} = u$. Furthermore, since the event $\mathsf{BadCall}_j$ occurs only if the first $j-1$ queries output `reject`, the only information about a_i, b_i that is available to the prover after making $j-1$ rejecting queries is the value $f_i = h_i^{a_i} g^{b_i}$ and the fact that $t^{a_i} \cdot s^{b_i} \notin U$ for some set U of size $|U| < j$. By Lemma 5.2, the values $t^{a_i} \cdot s^{b_i}$ and $h_i^{a_i} g^{b_i}$ are pairwise uniform in $\mathbb{G}$ over the randomness of a_i, b_i and so $t^{a_i} \cdot s^{b_i}$ is uniform over $G \setminus U$. This means that the probability of $\mathsf{BadCall}_j$ is $1/|G \setminus U|$ which is negligible.

Computational Hiding: We want to prove that, for all $I \subseteq [k]$, given:

$$\Big(\{(h_i, f_i = h_i^{a_i} g^{b_i}, \gamma)\}_{i \in [k]}, s = g^y, \{r_i = \mathsf{hc}(h_i^y; \gamma)\}_{i \in I}, \{\pi_i = (f_i^y,\, h_i^y)\}_{i \in I}\Big),$$

the values $\{r_i = \mathsf{hc}(h_i^y; \gamma)\}_{i \notin I}$ still look pseudorandom. Let $h_i = g^{x_i}$. Intuitively, the above follows since hc is a hard-core bit for $g^{x_i \cdot y}$ which is computationally unpredictable even given g^{x_i} and g^y. In more detail, we define a sequence of hybrids for $j \notin I$ where we switch $r_j = \mathsf{hc}(h_j^y; \gamma)$ to uniform one-by-one. To prove indistinguishability between any pair of successive hybrids, we give a reduction which receives γ, g^{x_j} and g^y along with a bit b such that either $b = \mathsf{hc}(g^{x_j y}; \gamma)$ or b is uniform. It samples itself a_i, b_i for all $i \in [k]$, x_i for all $i \neq j$, and computes $h_i = g^{x_i}$ and $f_i = g^{a_i x_i + b_i}$ for all $i \neq j$, and $f_j = (g^{x_j})^{a_j} \cdot g^{b_j}$. It computes itself for all $i \in I$: $r_i = \mathsf{hc}((g^y)^{x_i}; \gamma)$, $\pi_i = ((g^y)^{a_i x_i + b_i}, (g^y)^{x_i})$. Additionally, for $i \notin I, i < j$ it sets r_i as uniform and for $i \notin I, i > j$ it sets $r_i = \mathsf{hc}((g^y)^{x_i}; \gamma)$. Finally it sets $r_j = b$. Depending on whether $b = \mathsf{hc}(g^{x_j y}; \gamma)$ or b is uniform this corresponds to one of the two consecutive hybrids. $\square$

Combining Theorems 4.5 and 5.1, and Remark 2.5, we obtain the following:

Theorem 5.3 (Reusable DV-NIZK from CDH)**.** *Under the CDH assumption, there exists a reusable DV-NIZK for all NP with statistical soundness, and adaptive, multi-theorem zero-knowledge (Definitions 2.2, 4.2).*

6 Malicious-Designated-Verifier NIZKs

In this section we consider a strengthening of designated-verifier NIZKs to the malicious-designated-verifier setting (MDV-NIZK). In this setting, the trusted setup consists solely of a common random string (CRS). Given the CRS, the (potentially malicious) verifier generates a public key pk along with a secret key sk. The rest of the protocol is otherwise similar to the previous setting: any prover can use the CRS along with the newly generated public key to build non-interactive proofs of (many) NP statements, which can be verified using the corresponding secret key. The main difference is that we require zero-knowledge to hold against malicious verifiers, who can generate arbitrarily malformed public keys pk.

6.1 More Preliminaries

6.1.1 Reusable Malicious-Designated-Verifier NIZK

Definition 6.1 (Reusable Malicious-Designated-Verifier NIZK (MDV-NIZK))**.** *Let L be an NP language with witness relation R_L. A* Reusable Malicious-Designated-Verifier NIZK *(MDV-NIZK) for L is a tuple of PPT algorithms* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathcal{P}, \mathcal{V})$ *where:*

- $\mathsf{Setup}(1^\lambda, 1^n)$*: outputs a common random string* crs*;*
- $\mathsf{KeyGen}(\mathsf{crs})$*: outputs a public key* pk *along with an associated secret key* sk*;*
- $\mathcal{P}(\mathsf{crs}, \mathsf{pk}, x, w)$*: outputs a proof π;*
- $\mathcal{V}(\mathsf{crs}, \mathsf{sk}, \mathsf{pk}, x, \pi)$*: Outputs* `accept` *or* `reject`*.*

We require those algorithms to satisfy the same completeness and statistical soundness properties as Reusable DV-NIZKs (see Definition 2.2) with direct modifications to match the new syntax above, where now $(\mathsf{crs}, \mathsf{pk})$ *together act in place of what was previously just the* crs*. The requirement for zero-knowledge is strengthened to the following:*

Malicious Zero-Knowledge (Adaptive)*: We require that there exists a PPT simulator* Sim *such that for any PPT stateful adversary $\mathcal{A}$, the two following distributions are computationally indistinguishable:*

$\text{EXP}^{\mathit{Real}}(1^\lambda)$:	$\text{EXP}^{\mathit{Ideal}}(1^\lambda)$:
$1^n \leftarrow \mathcal{A}(1^\lambda)$	$1^n \leftarrow \mathcal{A}(1^\lambda)$
$\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$	$\mathsf{crs} \leftarrow \mathsf{Sim}(1^\lambda, 1^n)$
$(x, w, \mathsf{pk}) \leftarrow \mathcal{A}(\mathsf{crs})$	$(x, w, \mathsf{pk}) \leftarrow \mathcal{A}(\mathsf{crs})$
where $(x, w) \in R_L, \lvert x \rvert = n$	*where* $(x, w) \in R_L, \lvert x \rvert = n$
$\pi \leftarrow \mathcal{P}(\mathsf{crs}, \mathsf{pk}, x, w)$	$\pi \leftarrow \mathsf{Sim}(\mathsf{pk}, x)$
Output $\mathcal{A}(\pi)$	*Output* $\mathcal{A}(\pi)$

Remark 6.2 (Single-Theorem vs. Multi-Theorem Zero-Knowledge)**.** *As in Definition 2.2, the definition above only captures* single-theorem *zero-knowledge. However the same "Or trick" of [FLS99] as in Remark 2.5 allows to generically compile any MDV-NIZK with single-theorem, adaptive (resp. selective) ZK into one satisfying* multi-theorem*, adaptive (resp. selective) ZK.*

6.1.2 One-More CDH

We will use in this section a strengthening of the CDH assumption called *One-More CDH*. Intuitively, it states that given a set of challenge elements $\{h_j = g^{b_j}\}$ and the ability to make m queries to an oracle that raises arbitrary elements to some hidden exponent $a \in \mathbb{Z}_p$, it is hard to guess *more* than m of the values $h_j^{a_j} = g^{ab_j}$.

Definition 6.3 (One-More Computational Diffie-Hellman assumption (One-More CDH))**.** *Let* GroupGen *be a group generator. Let $\ell = \ell(\lambda)$ and $m = m(\lambda)$ be polynomials. Consider, for any PPT $\mathcal{A}$, the following experiment:*

$\mathsf{Exp}^{\textit{One-More CDH}}(1^\lambda)$

1. $(\mathbb{G}, p, g) \leftarrow \mathsf{GroupGen}(1^\lambda)$
2. $\left(g^a, \{g^{b_i}\}_{i\le\ell}\right) \xleftarrow{\$} \mathbb{G}^{1+\ell}$
3. $L \leftarrow \mathcal{A}^{\mathcal{O}_a(\cdot)}(\mathbb{G}, p, g, g^a, \{g^{b_i}\}_{i\le\ell})$
4. *Output* 1 *if* $\exists\, i_1 < \cdots < i_{m+1} \in [\ell]$ *such that* $\forall j \le m+1,, g^{a\cdot b_{i_j}} \in L$;
   Otherwise output 0,

where the oracle $\mathcal{O}_a$ takes as input a group element $h \in \mathbb{G}$ and outputs h^a.

We say that the One-More CDH *assumption holds relative to* GroupGen[7] *if for all PPT algorithm $\mathcal{A}$ making at most m queries to $\mathcal{O}_a$, we have:*

$$\Pr[\mathsf{Exp}^{\textit{One-More CDH}}(1^\lambda) = 1] \le \mathsf{negl}(\lambda).$$

Remark 6.4 (One-More CDH in Prior Works)**.** *A variety of previous works defined assumptions similar to the one above. To our knowledge, the first of this kind was introduced in the context of blind signatures in [Bol03], following the steps of [BNPS03] who first introduced One-More variants of the RSA and Discrete Log assumptions. More recently, another variant was used in the context of Oblivious PRFs (e.g. [JKK14]). The variant of [Bol03] requires the adversary to output one single guess for each target index $j \in J$, as opposed to a list of candidates L. As the adversary a-priori cannot test himself whether an element is correct, this makes it more difficult for the adversary to win the game and therefore the assumption of [Bol03] is weaker than our version in Definition 6.3. In [JKK14], on the other hand, the adversary is also given oracle access to a procedure that tests whether an element is a correct CDH output associated to some target index, but still has to output a single element for each target index. A direct reduction shows that this assumption is at least as strong as our variant: an adversary in the latter can call the oracle of [JKK14] on the whole list L to recover the matching indices.*

[7]Later, we will also use the (mild) additional property that one can *obliviously* sample uniform group elements in $\mathbb{G}$, so that the One-More CDH assumption holds even given the random coins used to sample the group elements in Step 2. (and in particular a and the b_i's should be computationally hidden). Note that most standard groups (such as $\mathbb{Z}_p^*$ or elliptic curves) allow to do so. Looking ahead, if such a property does not hold, the resulting MDV-NIZK (Theorem 6.17) will use a common *reference* string instead.

6.1.3 Somewhere-Equivocable PRFs (SEPRFs)

We recall here the concept of Somewhere-Equivocable pseudorandom function (SEPRF)s, introduced in [HJO+16]. This is a function $\mathsf{PRF}(K, \cdot)$ with two modes of generating a key. There is the standard key generation algorithm which generates a key K honestly. In addition, there is a way to generate a key K' that leaves a "hole" at some particular point x^* but defines the PRF output at all other points; later one can "plug the hole" to any value r by creating a key K^* which agrees with K' on all values other than x^* but on x^* it outputs r. For any x^* and a random r one cannot distinguish between an honestly generated key K and the key K^* created as above. Intuitively, the second mode of key generation ensures that the function $\mathsf{PRF}(K^*, \cdot)$ outputs a truly random and independent value on some specific point x^*.

Definition 6.5 (1-Somewhere-Equivocable PRFs (1-SEPRFs) [HJO+16])**.** *A* 1-Somewhere-Equivocable PRF *(1-SEPRF) with input size s and output size d is a tuple of PPT algorithms* $(\mathsf{ObvGen}, \mathsf{PRF}, \mathsf{Sim}_1, \mathsf{Sim}_2)$*:*

- $\mathsf{ObvGen}(1^\lambda)$*: outputs a key K such that* $\mathsf{PRF}(K, \cdot)$ *maps* $\{0,1\}^s$ *to* $\{0,1\}^d$*;*
- $\mathsf{Sim}_1(x^*)$*: on input $x^* \in \{0,1\}^s$, outputs a key K and a state* state*;*
- $\mathsf{Sim}_2(\mathsf{state}, r)$*: on input $r \in \{0,1\}^d$, outputs a key K'.*

such that the following properties hold:

Correctness: *We have that for all $x^* \in \{0,1\}^s$ and $r \in \{0,1\}^d$, if* $(K, \mathsf{state}) \xleftarrow{\$} \mathsf{Sim}_1(x^*)$ *and* $K' \xleftarrow{\$} \mathsf{Sim}_2(\mathsf{state}, r)$*, then:*

$$\begin{aligned} \mathsf{PRF}(K, x) &= \mathsf{PRF}(K', x) \qquad \text{if } x \neq x^* \\ \mathsf{PRF}(K', x^*) &= r. \end{aligned}$$

Equivocation security: *For all PPT adversary $\mathcal{A}$ we have:*

$$\left| \Pr\left[\begin{array}{c} x^* \xleftarrow{\$} \mathcal{A}(1^\lambda) \\ K \xleftarrow{\$} \mathsf{ObvGen}(1^\lambda) \\ \mathcal{A}(K) = 1 \end{array} \right] - \Pr\left[\begin{array}{c} x^* \xleftarrow{\$} \mathcal{A}(1^\lambda), r^* \xleftarrow{\$} \{0,1\}^d \\ (K, \mathsf{state}) \leftarrow \mathsf{Sim}_1(x^*) \\ K' \xleftarrow{\$} \mathsf{Sim}_2(\mathsf{state}, r^*) \\ \mathcal{A}(K') = 1 \end{array} \right] \right| \leq \mathsf{negl}(\lambda).$$

Claim 6.6 ([HJO+16])**.** *Assuming one-way functions exist, there exist 1-SEPRFs, with key size $\mathcal{O}(s \cdot d \cdot \lambda)$.*

6.2 Reusable Malicious-Designated-Verifier HBG (MDV-HBG)

To define a reusable Malicious-Designated-Verifier Hidden-Bits Generator (MDV-HBG), we extend the definition of a DV-HBG in a manner analogous to the difference between DV-NIZKs and MDV-NIZKs. Namely, instead of having a trusted setup that generates a public crs along with a secret key sk for the verifier, we now only have the setup algorithm generate the crs and allow the (potentially malicious) verifier to generate pk, sk on his own via a new KeyGen algorithm. Furthermore, we want to ensure that the generated hidden bits only depend on crs but not on pk; only the openings of the hidden bits can depend on pk.

Definition 6.7 (Reusable Malicious-Designated-Verifier HBG (MDV-HBG))**.** *A Reusable Malicious-Designated-Verifier HBG is a tuple of PPT algorithms* $(\mathsf{Setup}, \mathsf{KeyGen}, (\mathsf{GenBits.Commit}, \mathsf{GenBits.Prove}), \mathsf{Verify})$*:*

- $\mathsf{Setup}(1^\lambda, 1^k)$*: outputs a common random string* crs*.*
- $\mathsf{KeyGen}(\mathsf{crs})$*: outputs a public key* pk *with an associated secret key* sk*.*
- $\mathsf{GenBits}(\mathsf{crs}, \mathsf{pk})$ *is now split into two sub-procedures:*
  - $\mathsf{GenBits.Commit}(\mathsf{crs})$*: on input a* crs*, outputs a commitment* com*, some bits* $r \in \{0,1\}^k$ *and a state* state*.*
  - $\mathsf{GenBits.Prove}(\mathsf{crs}, \mathsf{pk}, \mathsf{state})$*: on input a public key* pk*, a* crs *and a state* state*, produces proofs* $\{\pi_i\}_{i \in k}$*.*

  It outputs $(\mathsf{com}, r, \{\pi_i\}_{i\in[k]})$*.*
- $\mathsf{Verify}(\mathsf{crs}, \mathsf{sk}, \mathsf{com}, i, r_i, \pi_i)$*: Outputs* `accept` *or* `reject`*.*

We require an MDV-HBG to satisfy the following properties. The first three (correctness, succinctness of the commitments and statistical binding), are direct adaptations of Definition 3.1 to the new syntax:

Correctness: *We require that for every polynomial $k = k(\lambda)$ and for all $i \in [k]$, we have:*

$$\Pr\left[\mathsf{Verify}(\mathsf{crs}, \mathsf{sk}, \mathsf{com}, i, r_i, \pi_i) = \mathtt{accept} \ : \ \begin{array}{cl} \mathsf{crs} & \leftarrow \mathsf{Setup}(1^\lambda, 1^k) \\ (\mathsf{pk}, \mathsf{sk}) & \leftarrow \mathsf{KeyGen}(\mathsf{crs}) \\ (\mathsf{com}, r, \pi_{[k]}) & \leftarrow \mathsf{GenBits}(\mathsf{crs}, \mathsf{pk}) \end{array}\right] = 1.$$

Succinct Commitment: *We require that there exists some set $\mathcal{COM}(\lambda)$ and some constant $\delta < 1$ such that $|\mathcal{COM}(\lambda)| \le 2^{k^\delta \mathsf{poly}(\lambda)}$, and such that for all* crs *output by* $\mathsf{Setup}(1^\lambda, 1^k)$ *and all* com *output by* $\mathsf{GenBits}(\mathsf{crs})$ *we have* $\mathsf{com} \in \mathcal{COM}(\lambda)$*. Furthermore, we require that for all* $\mathsf{com} \notin \mathcal{COM}(\lambda)$*,* $\mathsf{Verify}(\mathsf{crs}, \mathsf{com}, \cdot, \cdot)$ *always outputs* `reject`*.*

Statistical Binding: *There exists an (inefficient) deterministic algorithm* $\mathsf{Open}(1^k, \mathsf{crs}, \mathsf{com})$ *such that for every polynomial $k = k(\lambda)$, on input 1^k,* crs *and* com*, the algorithms outputs r such that for every (potentially unbounded) cheating prover $\widetilde{\mathcal{P}}$:*

$$\Pr\left[\begin{array}{cc} & r_i^* \neq r_i \\ \wedge & \mathsf{Verify}(\mathsf{crs}, \mathsf{sk}, \mathsf{com}, i, r_i^*, \pi_i) = \mathtt{accept} \end{array} \ : \ \begin{array}{cl} \mathsf{crs} & \leftarrow \mathsf{Setup}(1^\lambda, 1^k) \\ (\mathsf{pk}, \mathsf{sk}) & \leftarrow \mathsf{KeyGen}(\mathsf{crs}) \\ (\mathsf{com}, i, r_i^*, \pi_i) & \leftarrow \widetilde{\mathcal{P}}(\mathsf{crs}, \mathsf{pk}) \\ r & \leftarrow \mathsf{Open}(1^k, \mathsf{crs}, \mathsf{com}) \end{array}\right] \le \mathsf{negl}(\lambda).$$

The main conceptual difference with Definition 3.1 comes from the computational hiding property, which now captures security against malicious verifiers:

Computationally Hiding against Malicious Verifiers: *Consider, for an integer k, a bit b, and a stateful PPT adversary $\mathcal{A}$, the following experiment:*

```
$\mathsf{Exp}^{Hiding,b}(1^\lambda, 1^k)$
0. $I \subseteq [k] \leftarrow \mathcal{A}(1^k)$
1. $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^k)$
2. $\mathsf{pk} \leftarrow \mathcal{A}(\mathsf{crs})$
3. Compute $(\mathsf{com}, r, \{\pi_i\}_{i\in[k]}) \leftarrow \mathsf{GenBits}(\mathsf{crs}, \mathsf{pk})$.
   Set for all $i \notin I$ : $\begin{cases} \rho_i = r_i \text{ if } b = 0; \\ \rho_i \xleftarrow{\$} \{0,1\} \text{ otherwise.} \end{cases}$
4. Output : $\beta \leftarrow \mathcal{A}\left(\mathsf{crs}, \mathsf{com}, I, r_I, \pi_I, \{\rho_i\}_{i\notin I}\right)$
```

We require that for all polynomial $k = k(\lambda)$ and stateful PPT adversary $\mathcal{A}$:

$$\left|\Pr\left[\mathsf{Exp}^{Hiding,0}(1^\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{Hiding,1}(1^\lambda) = 1\right]\right| \le \mathsf{negl}(\lambda).$$

6.3 Reusable MDV-NIZK from MDV-HBG

We prove here an analogue to Theorem 4.1 in the malicious-verifier setting.

Theorem 6.8. *Suppose there exists a MDV-HBG. Then there exists a reusable MDV-NIZK with adaptive ZK security.*

The construction is a straight adaptation of the one in Section 4.1 to the new syntax, where the verifier's public key and secret key are respectively the public key and the secret key from the underlying HBG. The proofs of Correctness, Succinctness of the commitment and Statistical Binding are almost identical, with the only differences being syntactical (where $(\mathsf{crs}, \mathsf{pk})$ now acts as crs in the definitions and proofs from Sections 3 and 4).

The proof of Malicious Zero-Knowledge is also very similar to the proof of zero-knowledge in Section 4.1, where we now use the fact that $\mathsf{GenBits.Commit}$ does not take pk as an input. Intuitively this ensures that the adversary does not have any control on the generated hidden-bits. Technically, this allows the simulator to program as before the hidden-bits generated by the HBG to the simulated hidden-bits given by the hidden-bits NIZK simulator without the adversary being able to tell the difference.

Let us highlight the difference in the selective case for simplicity; the adaptive case being very similar. The only non-syntactical difference is that we can still switch from hybrid H_2 to hybrid H_3 (from the proof of zero-knowledge in Section 4.1), where in H_2 the advesary receives random hidden-bits, where in H_3 it receives simulated ones. These two hybrids are now slightly different syntactically:

$\mathbf{H_2}$: We now switch how s is computed.

- $r \xleftarrow{\$} \{0,1\}^k$
- $(I, \pi^{\mathsf{HB}}) \leftarrow \mathcal{P}^{\mathsf{HB}}(r, x, w)$

- $\mathsf{crs}^{\mathsf{BG}} \leftarrow \mathsf{Setup}^{\mathsf{BG}}(1^\lambda, 1^k), (\mathsf{com}, r^{\mathsf{BG}}, \mathsf{state}) \leftarrow \mathsf{GenBits.Commit}(\mathsf{crs}^{\mathsf{BG}})$
- $s_i = r_i \oplus r_i^{\mathsf{BG}}$ if $i \in I$ and $s_i \xleftarrow{\$} \{0,1\}$ otherwise.
- $\mathsf{pk} \leftarrow \mathcal{A}(\mathsf{crs} = (\mathsf{crs}^{\mathsf{BG}}, s))$
- $\pi_{[k]} \leftarrow \mathsf{GenBits.Prove}(\mathsf{crs}^{\mathsf{BG}}, \mathsf{pk}, \mathsf{state})$

Set $\mathsf{crs} = (\mathsf{crs}^{\mathsf{BG}}, s), \Pi = (I, \pi^{\mathsf{HB}}, \mathsf{com}, r_I, \pi_I)$, and output $\mathcal{A}(\mathsf{crs}, \Pi)$.

$\mathbf{H_3}$**:** We now switch how $(I, r_I, \pi^{\mathsf{HB}})$ are computed by using $\mathsf{Sim}^{\mathsf{HB}}(x)$ to generate them:

- $(I, r_I, \pi^{\mathsf{HB}}) \leftarrow \mathsf{Sim}^{\mathsf{HB}}(x)$
- $\mathsf{crs}^{\mathsf{BG}} \leftarrow \mathsf{Setup}^{\mathsf{BG}}(1^\lambda, 1^k)$, $(\mathsf{com}, r^{\mathsf{BG}}, \mathsf{state}) \leftarrow \mathsf{GenBits.Commit}(\mathsf{crs}^{\mathsf{BG}})$
- $s_i = r_i \oplus r_i^{\mathsf{BG}}$ if $i \in I$ and $s_i \xleftarrow{\$} \{0,1\}$ otherwise.
- $\mathsf{pk} \leftarrow \mathcal{A}(\mathsf{crs} = (\mathsf{crs}^{\mathsf{BG}}, s))$
- $\pi_{[k]} \leftarrow \mathsf{GenBits.Prove}(\mathsf{crs}^{\mathsf{BG}}, \mathsf{pk}, \mathsf{state})$

Set $\mathsf{crs} = (\mathsf{crs}^{\mathsf{BG}}, s), \Pi = (I, \pi^{\mathsf{HB}}, \mathsf{com}, r_I, \pi_I)$, and output $\mathcal{A}(\mathsf{crs}, \Pi)$.

Here we use the fact that the output of GenBits.Commit is independent of pk, so that we can compute s (included in the crs) *before* receiving the public key from the adversary. As before, the two hybrids are indistinguishable by zero-knowledge of the hidden-bits NIZK, and again, the view of the adversary in H_3 now corresponds to the one produced in the ideal experiment.

The difference in the adaptive case is almost identical: in the proof of Theorem 4.5, the same argument now allows to switch from $\mathbf{H_1}$ to $\mathbf{H_2}$.

6.4 MDV-HBG from One-More CDH

Notation. Let d, k and ℓ be integers, where ℓ is a power-of-two. Given a function $\varphi : [k] \to [\ell]^d$ and some index $i \in [k]$, we define, for some vector $\mathbf{u}$ of dimension ℓ, the vector:

$$\mathbf{u}_{\varphi(i)} := \left(u_{\varphi(i)_1}, \ldots, u_{\varphi(i)_d}\right) .$$

In other words, we can think of $\varphi(i)$ as a set of *neighbors* of vertex $i \in [k]$ in the bipartite (multi-)graph $([k], [\ell])$. Furthermore if the vertices $j \in [\ell]$ are labelled with some element u_j, then $u_{\varphi(i)}$ denotes the *list* of labels associated to neighbors of i. Note that vertices in $[k]$ have d neighbors in $[\ell]$ (where there can be multiple occurrences of the same edge). We naturally extend this definition for *sets* of indices: for $I \subseteq [k]$, we define

$$\mathbf{u}_{\varphi(I)} := \left(u_{\varphi(i)_1}, \ldots, u_{\varphi(i)_d}\right)_{i \in I} .$$

Let hc be the Goldreich-Levin [GL89] hard-core bit (which, on input a bit-string $x \in \{0,1\}^L$, uses randomness $r \xleftarrow{\$} \{0,1\}^L$ and outputs $\mathsf{hc}(x; r) := (\langle x, r\rangle, r))$.

Construction. Let $(\mathbb{G}, p, g) \leftarrow \mathsf{GroupGen}(1^\lambda)$ be a prime-order group generator so that $\mathbb{G}$ is a group of prime order p, with a generator g. For $\lambda, k \in \mathbb{N}$, let $\ell = \ell(\lambda, k)$ be the least power-of-two greater than $3k\lambda$ (i.e. $\ell = 2^{\lceil \log(3k\lambda) \rceil}$), and let $d = \lambda$. Let $(\mathsf{ObvGen}, \mathsf{PRF}, \mathsf{Sim}_1, \mathsf{Sim}_2)$ be a 1-SEPRF (as defined in Section 6.1.3) where $\mathsf{ObvGen}(1^\lambda)$ outputs keys K such that $\mathsf{PRF}(K, \cdot)$ maps $\{0,1\}^{\lceil \log k \rceil}$ to $\{0,1\}^{d \cdot \log \ell}$ (and in particular maps $[k]$ to $[\ell]$).

Let us define the following hidden-bits generator:

- $\mathsf{Setup}(1^\lambda, 1^k)$: Let $(\mathbb{G}, p, g) \leftarrow \mathsf{GroupGen}(1^\lambda)$. For all $j \in [\ell]$, pick $h_j \xleftarrow{\$} \mathbb{G}$. Output:

$$\mathsf{crs} = (\mathbb{G}, \{h_j\}_{j \in [\ell]}).$$

- $\mathsf{KeyGen}(\mathsf{crs})$: For all $j \in [\ell]$, pick random $a_j, b_j \xleftarrow{\$} \mathbb{Z}_p$, compute:

$$f_j = h_j^{a_j} \cdot g^{b_j},$$

  and output:

$$\mathsf{pk} = \{f_j\}_{j \in [\ell]},$$
$$\mathsf{sk} = \{(a_j, b_j)\}_{j \in [\ell]}.$$

- $\mathsf{GenBits}(\mathsf{crs}, \mathsf{pk})$:

  - $\mathsf{GenBits.Commit}(\mathsf{crs})$: Pick a random $y \leftarrow \mathbb{Z}_p$ and set $s = g^y$. Compute for all $j \in [\ell]$: $t_j = h_j^y$. Sample some random coins γ matching the randomness used by $\mathsf{hc}(\cdot)$ taking as input (the bit-representation of) elements in $\mathbb{G}^d$. Sample $K \leftarrow \mathsf{ObvGen}(1^\lambda)$. Parsing the output of $\mathsf{PRF}(K, \cdot)$ as d blocks of $\log \ell$ bits, this defines for all $i \in [k]$:

$$\varphi(i) := (\mathsf{PRF}(K, i)_1, \ldots, \mathsf{PRF}(K, i)_d) \in [\ell]^d. \qquad (1)$$

    Compute for all $i \in [k]$: $r_i = \mathsf{hc}\left((\mathbf{h}^y)_{\varphi(i)}\,;\,\gamma\right)$, where we recall that by definition $(\mathbf{h}^y)_{\varphi(i)} = \left(h^y_{\mathsf{PRF}(K,i)_1}, \ldots, h^y_{\mathsf{PRF}(K,i)_d}\right)$. Output:

$$\mathsf{com} = (s, \gamma, K),$$
$$\{r_i\}_{i \in [k]},$$
$$\mathsf{state} = (y, K).$$

  - $\mathsf{GenBits.Prove}(\mathsf{crs}, \mathsf{pk}, \mathsf{state})$: Parse pk as $\{f_j\}_{j \in [\ell]}$. The key K in state defines a function φ as per Equation 1. Compute for all $j \in [\ell]$: $t_j = h_j^y$ and $u_j = f_j^y$. Compute for all $i \in [k]$:

$$\pi_i = \{(t_j, u_j)\}_{j \in \varphi(i)}.$$

  Output:

$$(\mathsf{com}, r, \{\pi_i\}_{i \in [k]}).$$

- $\mathsf{Verify}(\mathsf{crs}, \mathsf{sk}, \mathsf{com}, i, r_i, \pi_i)$: Parse $\mathsf{sk} = \{(a_j, b_j)\}_{j\in[\ell]}, \mathsf{com} = (s, \gamma, K), \pi_i = \{(t_j, u_j)\}_{j\in\varphi(i)}$. Compute for $j \in \varphi(i)$ (where $\varphi(i)$ is defined as per Equation 1):

$$\rho_j = t_j^{a_j} \cdot s^{b_j},$$

and accept if and only if $\rho_j = u_j$ for all $j \in \varphi(i)$, and $r_i = \mathsf{hc}\left(\{\mathbf{t}\}_{\varphi(i)} \ ; \gamma\right)$.

Theorem 6.9. *Suppose that* $(\mathsf{ObvGen}, \mathsf{PRF}, \mathsf{Sim}_1, \mathsf{Sim}_2)$ *is a 1-SEPRF (Definition 6.5). Then, assuming the One-More CDH assumption holds (Definition 6.3,* $(\mathsf{Setup}, \mathsf{GenBits}, \mathsf{Verify})$ *is a reusable Malicious-Designated-Verifier Hidden-Bits Generator (Definition 6.7).*

Claim 6.10. $(\mathsf{Setup}, \mathsf{GenBits}, \mathsf{Verify})$ *satisfy completeness, statistical binding and succinctness of its commitments (Definition 6.7).*

Proof. Completeness and statistical binding are direct adaptations of the corresponding proof of Theorem 5.1, where we now define:

- $\mathsf{Open}(1^k, \mathsf{crs}, \mathsf{com})$: Parse com as (s, γ, K) (which defines φ by Equation 1). Compute (inefficiently) $y \in \mathbb{Z}_p$ such that $g^y = s$, and output:

$$r_i = \mathsf{hc}\left((\mathbf{h}^y)_{\varphi(i)} \ ; \gamma\right).$$

For succinctness, we use the fact that both the bit-length of γ (used by hc) and the number of SEPRF keys are independent of k, and polynomial in λ. This is because the group elements size and the size d of the (multi)sets $\varphi(i)$ satisfy those properties. The (bit-)size of the SEPRF keys grows *logarithmically* with k (as its output consists of $\log k$ bits) (and polynomially with λ). Overall the set of valid commitments is of size at most $\mathsf{poly}(k) \cdot 2^{\mathsf{poly}(sec)}$, which gives succinctness. □

It remains to prove computationally hiding against malicious verifiers as defined in Section 6.2, which is the most technical part of the proof.

Claim 6.11. *Assume that* $(\mathsf{ObvGen}, \mathsf{PRF}, \mathsf{Sim}_1, \mathsf{Sim}_2)$ *is a 1-SEPRF (Definition 6.5). Then, assuming the One-More CDH assumption holds (Definition 6.3), then* $(\mathsf{Setup}, \mathsf{GenBits}, \mathsf{Verify})$ *is computationally hiding against malicious verifiers (Definition 6.7).*

6.4.1 Proof of Claim 6.11

Proof. Let $I \subseteq [k]$ be the set of indices fixed by the adversary in the malicious computational hiding experiment. Without loss of generality, we can suppose $I = [k] \setminus \{i^*\}$ for some index $i^* \in [k]$. This is because if malicious computational hiding holds for all such sets, then it also holds for all subsets $I \subseteq [k]$ by a simple hybrid argument.

The proof goes in three steps. First, we show that any distinguisher breaking malicious hiding of the HBG induces an appropriate *decoder* (Definition 6.12, Lemma 6.13). Then, we show that any such decoder also succeeds in a slightly modified decoding experiment (Definition 6.14, Lemma 6.15). Finally, we turn any such decoder into an adversary for the One-More CDH experiment (Definition 6.3, Lemma 6.16).

We first introduce a bit of notation, and define the following modified commit algorithm which we will use in our decoding experiment:

- $\mathsf{GenBits.Commit}^{\mathsf{Dec}}(\mathsf{crs})$:
  - First, compute $\mathsf{GenBits.Commit}_1(\mathsf{crs}) = (s = g^y, \mathsf{state}_1 = y)$;
  - Second, compute $\mathsf{GenBits.Commit}_2(\mathsf{crs}, \mathsf{state}_1) = (K, \mathsf{state}_2 = (y, K))$.

  Output:
  $$\mathsf{com}^{\mathsf{Dec}} = (s, K), \quad \mathsf{state} = (y, K).$$

Namely, the algorithm $\mathsf{GenBits.Commit}^{\mathsf{Dec}}$ does not sample the randomness γ for the hard-core bit, and does not compute the hidden-bits r.

This induces the following modified algorithm:

- $\mathsf{GenBits}^{\mathsf{Dec}}(\mathsf{crs}, \mathsf{pk})$:
  - Compute $(\mathsf{com}^{\mathsf{Dec}}, \mathsf{state}) \leftarrow \mathsf{GenBits.Commit}^{\mathsf{Dec}}(\mathsf{crs})$;
  - Compute $\{\pi_i\}_{i\in[k]} \leftarrow \mathsf{GenBits.Prove}(\mathsf{crs}, \mathsf{pk}, \mathsf{state})$.

  Output:
  $$(\mathsf{com}^{\mathsf{Dec}}, \{\pi_i\}_{i\in[k]}),$$

We now describe our (first) decoding experiment:

Definition 6.12 (Decoding Experiment)**.** *Consider, for an integer k, and a stateful adversary $\mathcal{A}$, the following experiment:*

$\mathsf{Exp}^{Dec}(1^\lambda, 1^k)$

0. $I = [k] \setminus \{i^*\} \leftarrow \mathcal{A}(1^k)$
1. $\mathsf{crs} = (\mathbb{G}, \{h_j\}_{j\in[\ell]}) \leftarrow \mathsf{Setup}(1^\lambda, 1^k)$
2. $\mathsf{pk} \leftarrow \mathcal{A}(\mathsf{crs})$
3. *Compute* $(\mathsf{com}^{\mathsf{Dec}} = (s = g^y, K), \{\pi_i\}_{i\in[k]}) \leftarrow \mathsf{GenBits}^{\mathsf{Dec}}(\mathsf{crs}, \mathsf{pk})$.
4. $\mathbf{x} \in \mathbb{G}^d \leftarrow \mathcal{A}\left(\mathsf{crs}, \mathsf{com}^{\mathsf{Dec}}, \pi_I\right)$.

$$\textit{Output} \ : \begin{cases} 1 \ \textit{if}\ \mathbf{x} = (\mathbf{h}^y)_{\varphi(i^*)} \\ 0 \ \textit{otherwise} \end{cases},$$

where φ is defined by K as in Equation 1.

We say that a PPT adversary $\mathcal{A}$ is a decoder for $\mathsf{Exp}^{Dec}(1^\lambda, 1^k)$ if there exists a non-negligible function α such that:

$$\Pr[\mathsf{Exp}^{Dec}(1^\lambda, 1^k)) = 1] \geq \alpha(\lambda).$$

Lemma 6.13 (Distinguisher to decoder [GL89])**.** *Suppose hc is the Goldreich-Levin hard-core bit [GL89]. Suppose furthermore that there exists an efficient distinguisher for the malicious hiding experiments $\mathsf{Exp}^{Hiding,b}(1^\lambda, 1^k)$ (Definition 6.7). Then there exists a PPT decoder for $\mathsf{Exp}^{Dec}(1^\lambda, 1^k)$.*

The proof of Lemma 6.13 is almost identical to the one of the (plain) decoding property of the Goldreich-Levin hard-core bit [GL89]. Namely, we only use the fact that for all γ (which is picked by the reduction in the proof), the bits r_I are efficiently computable given γ and π_I (which are provided by the decoding experiment).

Next, we slightly modify the decoding experiment of Definition 6.12, namely, we modify the way $\mathsf{GenBits.Commit}^{\mathsf{Dec}}$ computes SEPRF keys. Recall that the SEPRF consists of algorithms $(\mathsf{ObvGen}, \mathsf{PRF}, \mathsf{Sim}_1, \mathsf{Sim}_2)$ (Section 6.1.3). Define the following modified $\mathsf{GenBits.CommitSim}^{\mathsf{Dec}}$ algorithm:

- $\mathsf{GenBits.CommitSim}^{\mathsf{Dec}}(\mathsf{crs}, I = [k] \setminus \{i^*\})$:
  - First, compute $\mathsf{GenBits.CommitSim}_1(\mathsf{crs}) = (s = g^y, \mathsf{state}_1 = y)$;
  - Second, compute $\mathsf{GenBits.CommitSim}_2(\mathsf{crs}, I)$ as follows:
    1. $\mathsf{GenBits.CommitSim}_2^{(1)}(\mathsf{crs})$: Compute $(K, \mathsf{state}) \leftarrow \mathsf{Sim}_1(i^*)$.
       Note that at the point one can compute $\pi_i \leftarrow \mathsf{GenBits.Prove}(\mathsf{crs}, \mathsf{com})$ *for all* $i \neq i^*$, by correctness of the SEPRF.
    2. $\mathsf{GenBits.CommitSim}_2^{(2)}(\mathsf{crs}, \mathsf{state})$: Sample a random $\rho \xleftarrow{\$} [\ell]^d$ and output $K' \leftarrow \mathsf{Sim}_2(\rho, \mathsf{state})$.

  Output:
$$\mathsf{comDec} = (s, K'), \quad \mathsf{state} = (y, K').$$

This naturally defines an algorithm $\mathsf{GenBitsSim}^{\mathsf{Dec}}(\mathsf{crs}, \mathsf{pk}, I)$ which runs $\mathsf{GenBits.CommitSim}^{\mathsf{Dec}}(\mathsf{crs}, I)$ (instead of $\mathsf{GenBits.Commit}^{\mathsf{Dec}}(\mathsf{crs})$), and the original algorithm $\mathsf{GenBits.Prove}(\mathsf{crs}, \mathsf{pk}, \mathsf{state})$. This in turn, defines the following natural modified decoding experiment:

Definition 6.14 (Simulated decoding experiment)**.** *Consider, for an integer k, and a stateful adversary $\mathcal{A}$, the following experiment:*

$\mathsf{Exp}_{Sim}^{Dec}(1^\lambda, 1^k)$

0. $I = [k] \setminus \{i^*\} \leftarrow \mathcal{A}(1^k)$
1. $\mathsf{crs} = (\mathbb{G}, \{h_j\}_{j \in [\ell]}) \leftarrow \mathsf{Setup}(1^\lambda, 1^k)$
2. $\mathsf{pk} \leftarrow \mathcal{A}(\mathsf{crs})$
3. *Compute* $(\mathsf{com}^{\mathsf{Dec}} = (s = g^y, K'), \{\pi_i\}_{i \in [k]}) \leftarrow \mathsf{GenBitsSim}^{\mathsf{Dec}}(\mathsf{crs}, \mathsf{pk}, I)$.
4. $\mathbf{x} \in \mathbb{G}^d \leftarrow \mathcal{A}\left(\mathsf{crs}, \mathsf{com}^{\mathsf{Dec}}, \pi_I\right)$.

$$\textit{Output} \ : \begin{cases} 1 \ \textit{if}\ \mathbf{x} = (\mathbf{h}^y)_{\varphi(i^*)} \\ 0 \ \textit{otherwise} \end{cases} .$$

We say that a PPT adversary $\mathcal{A}$ is a decoder for $\mathsf{Exp}_{Sim}^{Dec}(1^\lambda, 1^k)$ if there exists a non-negligible function α such that:

$$\Pr[\mathsf{Exp}^{Dec}(1^\lambda, 1^k)) = 1] \geq \alpha(\lambda).$$

Now the outputs of $\mathsf{Exp}^{\mathsf{Dec}}(1^\lambda, 1^k)$ and $\mathsf{Exp}_{Sim}^{\mathsf{Dec}}(1^\lambda, 1^k)$, with the same adversary $\mathcal{A}$, are computationally indistinguishable; this follows directly by the equivocation property of the SEPRF (Definition 6.5).

Lemma 6.15. *Suppose* $(\mathsf{ObvGen}, \mathsf{PRF}, \mathsf{Sim}_1, \mathsf{Sim}_2)$ *is a 1-SEPRF (Definition 6.5). Then, for all PPT adversary* $\mathcal{A}$*:*

$$\left|\Pr[\mathsf{Exp}^{Dec}(1^\lambda, 1^k) = 1] \;-\; \Pr[\mathsf{Exp}^{Dec}_{Sim}(1^\lambda, 1^k) = 1]\right| \leq \mathsf{negl}(\lambda).$$

Finally, we argue that any decoder for $\mathsf{Exp}^{\mathrm{Dec}}_{Sim}$ induces an adversary for the One-More CDH experiment:

Lemma 6.16. *Suppose that there exists an efficient decoder* $\mathcal{A}$ *for* $\mathsf{Exp}^{Dec}_{Sim}(1^\lambda, 1^k)$ *(Definition 6.14), and that* $(\mathsf{ObvGen}, \mathsf{PRF}, \mathsf{Sim}_1, \mathsf{Sim}_2)$ *satisfies correctness (Definition 6.5). Then there exists an efficient adversary for the One-More CDH experiment (Definition 6.3) with non-negligible advantage.*

Given such an efficient decoder $\mathcal{A}$ for $\mathsf{Exp}^{\mathrm{Dec}}_{Sim}(1^\lambda, 1^k)$, we build an adversary for the One-More CDH experiment as follows:

Adversary for the One-More CDH experiment: Let $\mathcal{A}$ be a decoder for $\mathsf{Exp}^{\mathrm{Dec}}_{Sim}(1^\lambda, 1^k)$ with non-negligible advantage $\alpha(\lambda)$. Define the following adversary for the One-More CDH experiment $\mathsf{Exp}^{\text{One-More CDH}}(1^\lambda)$:

1. First initialize an empty set L, and run the decoder $\mathcal{A}$ to define the index set $I = [k] \setminus \{i^*\}$.
2. After receiving group elements $\{h_j\}_{j\in[\ell]}$, and g^y from Step 2. of the One-More CDH experiment $\mathsf{Exp}^{\text{One-More CDH}}(1^\lambda)$, it sets

   $$\mathsf{crs} = \{h_j\}_{j\in[\ell]}, \quad s = g^y,$$

   (which is normally output by $\mathsf{GenBits.CommitSim}_1(\mathsf{crs})$). It then runs the decoder $\mathcal{A}$ to get a public key: $\mathsf{pk} = \{f_j\}_{j\in[\ell]} \leftarrow \mathcal{A}(\mathsf{crs})$.

   Then, it computes $(K, \mathsf{state}) \leftarrow \mathsf{Sim}_1(i^*)$ using the 1-SEPRF simulator (which is normally the output of $\mathsf{GenBits.CommitSim}_2^{(1)}(\mathsf{crs})$). Note that by correctness of the SEPRF, the simulated key K defines φ everywhere except on i^* as per Equation 1.

   It now submits $2|I|d = 2(k-1)d$ CDH oracle queries $(\mathbf{h}_{\varphi(I)}, \mathbf{f}_{\varphi(I)})$ using its knowledge of crs, pk and $\varphi(I)$. It receives the values $(\mathbf{h}^y_{\varphi(I)}, \mathbf{f}^y_{\varphi(I)})$, which he sets as π_I (normally output of $\mathsf{GenBits.Prove}$).

   It also adds the elements of $\{\mathbf{h}^y\}_{\varphi(I)}$ to the list L.
3. Repeat the following polynomially many times (discussed later):
   - Sample a random value $\rho \xleftarrow{\$} [\ell]^d$. Compute $K' \leftarrow \mathsf{Sim}_2(\rho, \mathsf{state})$, and set $(\mathsf{com}^{\mathsf{Dec}} = (s, K')$.
   - Run the decoder: $L' \leftarrow \mathcal{A}(\mathsf{crs}, \mathsf{com}^{\mathsf{Dec}}, \pi_I)$ (from Step 4 of $\mathsf{Exp}^{\mathrm{Dec}}_{Sim}(1^\lambda, 1^k)$), and all the elements of L' to L.
4. Output the list L.

To argue that this adversary indeed breaks the One-More CDH assumption, it suffices to show that:

- With good probability, if the adversary runs Step 3 sufficiently many times, then L will eventually succeed many times and include many sets $\mathbf{h}^y_{\varphi(i^*)}$ (which are defined by Steps 1 to 3);
- With good probability, those sets $\mathbf{h}^y_{\varphi(i^*)}$ include new CDH elements that were not previously in L.

Let $\mathsf{coins} = (\mathsf{coins}_1, \mathsf{coins}_2)$ be the randomness used by our adversary, where:

- coins_1 denotes the random coins used in Step 2. In the decoding experiment, this corresponds to all the randomness used by Setup, $\mathsf{GenBits.CommitSim}_1$, $\mathsf{GenBits.CommitSim}_2^{(1)}(\mathsf{crs})$ and $\mathsf{GenBits.Prove}$.
- coins_2 denotes the random coins used in Step 3. In the decoding experiment, this corresponds to the randomness used by $\mathsf{GenBits.CommitSim}_2^{(2)}(\mathsf{crs}, \mathsf{state})$, and the randomness of the decoder at Step 4. of $\mathsf{Exp}^{\mathrm{Dec}}_{Sim}(1^\lambda, 1^k)$.

We say that coins_1 are *good*, and write $\mathsf{coins}_1 \in \mathsf{Good}$, if:

$$\Pr_{\mathsf{coins}_2}\left[\mathsf{Exp}^{\mathrm{Dec}}_{Sim}(1^\lambda, 1^k; (\mathsf{coins}_1, \mathsf{coins}_2)) = 1\right] \geq \alpha(\lambda)/2.$$

A standard averaging argument (over the success probability of $\mathcal{A}$ in $\mathsf{Exp}^{\mathrm{Dec}}_{Sim}(1^\lambda, 1^k)$) shows that $\Pr[\mathsf{coins}_1 \in \mathsf{Good}] \geq \alpha(\lambda)/2$, and is in particular non-negligible.

Let L be the current list of group elements; let us analyze the behaviour of Step 3. Suppose without loss of generality that L contains at most $2(k-1)d$ distinct elements $\{h^y_j\}_{j\in[\ell]}$; otherwise L already makes the One-More CDH experiment output 1. Let $\mathsf{known} \subseteq [\ell]$ be a set of indices defined as:

$$i \in \mathsf{known} \iff h^y_i \in L.$$

Then, if the decoder succeeds, $\mathbf{h}^y_{\varphi(i^*)}$ contains no new CDH elements in L only if $\varphi(i^*) \subseteq \mathsf{known}$, that is $\rho_l \in \mathsf{known}$ for all $l \in [d]$, by definition of φ and correctness of the SEPRF. Now ρ is entirely determined by coins_2; and over the randomness of coins_2, the above happens with probability at most $(|\mathsf{known}|/\ell)^d$, which is negligible by our choice of parameters.

In other words, define the set Bad as $\mathsf{coins}_2 \in \mathsf{Bad}$ if, for all $l \in [d]$, $\rho_l \in \mathsf{known}$. Now, the probability that the decoder $\mathcal{A}$ succeeds in Step 3. and $\mathsf{coins}_2 \notin \mathsf{Bad}$ is at least $\mathsf{Adv} = \alpha(\lambda) - \mathsf{negl}(\lambda)$ which is non-negligible, independently of the list L (as long as it does not contain enough elements). Now:

- We have: $\Pr[\mathsf{coins}_1 \in \mathsf{Good}] \geq \alpha(\lambda)/2$;
- Suppose $\mathsf{coins}_1 \in \mathsf{Good}$. Let q be a polynomial such that $\mathsf{Adv} \geq 1/q(\lambda)$ for infinitely many λ. Then, if the adversary runs Step 3. $\lambda \cdot q(\lambda)$ times, then for such λ, we have that with overwhelming probability, it adds a correct, new distinct element h^y_i to L (supposing L does not contain enough elements yet).

In particular if we run Step 3. $\lambda \cdot q(\lambda) \cdot (|I| \cdot d + 1)$ times, then with overwhelming probability (for such λ), the list L will have at least $2(k-1)d+1$ distinct CDH elements.

Overall, our adversary wins in the the One-More CDH experiment with probability $\alpha(\lambda)/2 - \mathsf{negl}(\lambda)$ for infinitely many λ.

Wrapping-up. Assuming a decoder for $\mathsf{Exp}^{\mathrm{Dec}}_{Sim}(1^\lambda, 1^k)$, we built a PPT adversary for the One-More CDH experiment $\mathsf{Exp}^{\text{One-More CDH}}(1^\lambda)$ such that:

- It uses (at most) $2|I| \cdot d = 2(k-1) \cdot d$ calls to the CDH oracle (in Step 2);
- It outputs a list of group elements L such that with non-negligible probability, there are at least $2(k-1) \cdot d + 1$ indices $i \in [\ell]$, such that L contains all the associated CDH elements h_i^y.

Therefore such an adversary breaks the One-More CDH assumption This concludes the proof of Lemma 6.16, and therefore the proof of Claim 6.11. □

Combining Claim 6.6, Theorems 6.8 and 6.9, and Remark 6.2, we obtain the following:

Theorem 6.17 (MDV-NIZK from One-More CDH)**.** *Under the One-More CDH assumption (Definition 6.3), there exists a MDV-NIZK for all NP (Definition 6.1) with statistical soundness, and adaptive, multi-theorem zero-knowledge.*

7 Extensions

We informally describe two simple extensions of our construction.

Unbounded Statement Size. In our construction of (reusable DV-)NIZKs, we need to have a bound n on the size of the statements that can be proved and the size of the CRS depends on n. Ideally, we would have a fixed-size CRS which allows us to prove statements of arbitrary size. Indeed, we can achieve this using non-interactive statistically-binding commitments in the CRS model, which exist assuming OWFs [Nao90, Nao91]. Let us fix 3SAT as the NP-complete language. To prove that some 3CNF is satisfiable the prover commits to the satisfying assignments one variable at a time. Then he uses a (reusable DV-)NIZK scheme for each clause separately to show that the 3 relevant committed values satisfy the clause. Note that the size of the statements being proved by the underlying (reusable DV-)NIZK is independent of the size of the actual 3CNF formula. Therefore the above technique bootstraps a (reusable DV-)NIZK for statements of some fixed size which depends only on the security parameter to construct a (reusable DV-)NIZK for statements of arbitrary size.

Proof of Knowledge. While our basic construction is not a proof-of-knowledge it is easy to generically add this property assuming the existence of public-key encryption (PKE). We can add a public-key com of a PKE scheme to the CRS and have the prover encrypt the witness under com and then use the (reusable DV-)NIZK to prove that the ciphertext is an encryption of a valid witness for the statement. The extractor would choose com along with a corresponding decryption key sk and use it to extract the witness.

Acknowledgments

We thank Geoffroy Couteau, Dennis Hofheinz, Shuichi Katsumata, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa for sharing their manuscripts [CH19, KNYY19] and for helpful discussions.

References

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *20th Annual ACM Symposium on Theory of Computing*, pages 103–112, Chicago, IL, USA, May 2–4, 1988. ACM Press.

[BGI15] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 337–367, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.

[BNPS03] Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. *Journal of Cryptology*, 16(3):185–215, June 2003.

[Bol03] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Yvo Desmedt, editor, *PKC 2003: 6th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 31–46, Miami, FL, USA, January 6–8, 2003. Springer, Heidelberg, Germany.

[BY93] Mihir Bellare and Moti Yung. Certifying cryptographic tools: The case of trapdoor permutations. In Ernest F. Brickell, editor, *Advances in Cryptology – CRYPTO'92*, volume 740 of *Lecture Notes in Computer Science*, pages 442–460, Santa Barbara, CA, USA, August 16–20, 1993. Springer, Heidelberg, Germany.

[CC18] Pyrros Chaidos and Geoffroy Couteau. Efficient designated-verifier non-interactive zero-knowledge proofs of knowledge. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part III*, volume 10822 of *Lecture Notes in Computer Science*, pages 193–221, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany.

[CCRR18] Ran Canetti, Yilei Chen, Leonid Reyzin, and Ron D. Rothblum. Fiat-Shamir and correlation intractability from strong KDM-secure encryption. In *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part I*, pages 91–122, 2018.

[CH19] Geoffroy Couteau and Dennis Hofheinz. Towards non-interactive zero-knowledge proofs from CDH and LWE. In *EUROCRYPT*, 2019.

[CHK03] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 255–271, Warsaw, Poland, May 4–8, 2003. Springer, Heidelberg, Germany.

[CKS08] David Cash, Eike Kiltz, and Victor Shoup. The twin Diffie-Hellman problem and applications. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 127–145, Istanbul, Turkey, April 13–17, 2008. Springer, Heidelberg, Germany.

[CL17] Ran Canetti and Amit Lichtenberg. Certifying trapdoor permutations, revisited. *IACR Cryptology ePrint Archive*, 2017:631, 2017.

[CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25, Santa Barbara, CA, USA, August 23–27, 1998. Springer, Heidelberg, Germany.

[CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 45–64, Amsterdam, The Netherlands, April 28 – May 2, 2002. Springer, Heidelberg, Germany.

[Dam93] Ivan Damgård. Non-interactive circuit based proofs and non-interactive perfect zero-knowledge with proprocessing. In Rainer A. Rueppel, editor, *Advances in Cryptology – EUROCRYPT'92*, volume 658 of *Lecture Notes in Computer Science*, pages 341–355, Balatonfüred, Hungary, May 24–28, 1993. Springer, Heidelberg, Germany.

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *23rd Annual ACM Symposium on Theory of Computing*, pages 542–552, New Orleans, LA, USA, May 6–8, 1991. ACM Press.

[DFN06] Ivan Damgård, Nelly Fazio, and Antonio Nicolosi. Non-interactive zero-knowledge from homomorphic encryption. In Shai Halevi and Tal Rabin, editors, *TCC 2006: 3rd Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 41–59, New York, NY, USA, March 4–7, 2006. Springer, Heidelberg, Germany.

[DMP88] Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Non-interactive zero-knowledge proof systems. In Carl Pomerance, editor, *Advances in Cryptology – CRYPTO'87*, volume 293 of *Lecture Notes in Computer Science*, pages 52–72, Santa Barbara, CA, USA, August 16–20, 1988. Springer, Heidelberg, Germany.

[DMP90] Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Non-interactive zero-knowledge with preprocessing. In Shafi Goldwasser, editor, *Advances in Cryptology – CRYPTO'88*, volume 403 of *Lecture Notes in Computer Science*, pages 269–282, Santa Barbara, CA, USA, August 21–25, 1990. Springer, Heidelberg, Germany.

[DN00] Cynthia Dwork and Moni Naor. Zaps and their applications. In *41st Annual Symposium on Foundations of Computer Science*, pages 283–293, Redondo Beach, CA, USA, November 12–14, 2000. IEEE Computer Society Press.

[FLS99] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple noninteractive zero knowledge proofs under general assumptions. *SIAM J. Comput.*, 29(1):1–28, September 1999.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, Santa Barbara, CA, USA, August 1987. Springer, Heidelberg, Germany.

[GI14] Niv Gilboa and Yuval Ishai. Distributed point functions and their applications. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 640–658, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *21st Annual ACM Symposium on Theory of Computing*, pages 25–32, Seattle, WA, USA, May 15–17, 1989. ACM Press.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th Annual ACM Symposium on Theory of Computing*, pages 291–304, Providence, RI, USA, May 6–8, 1985. ACM Press.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[Gol01] Oded Goldreich. *Foundations of Cryptography: Basic Tools*, volume 1. Cambridge University Press, Cambridge, UK, 2001.

[Gol04] Oded Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004.

[Gol11] Oded Goldreich. *Basing Non-Interactive Zero-Knowledge on (Enhanced) Trapdoor Permutations: The State of the Art*, pages 406–421. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 339–358, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Heidelberg, Germany.

[GR13] Oded Goldreich and Ron D. Rothblum. Enhancements of trapdoor permutations. *J. Cryptology*, 26(3):484–512, 2013.

[HJO$^+$16] Brett Hemenway, Zahra Jafargholi, Rafail Ostrovsky, Alessandra Scafuro, and Daniel Wichs. Adaptively secure garbled circuits from one-way functions. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part III*, volume 9816 of *Lecture Notes in Computer Science*, pages 149–178, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany.

[HRW16] Dennis Hofheinz, Vanishree Rao, and Daniel Wichs. Standard security does not imply indistinguishability under selective opening. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B: 14th Theory of Cryptography Conference, Part II*, volume 9986 of *Lecture Notes in Computer Science*, pages 121–145, Beijing, China, October 31 – November 3, 2016. Springer, Heidelberg, Germany.

[JKK14] Stanislaw Jarecki, Aggelos Kiayias, and Hugo Krawczyk. Round-optimal password-protected secret sharing and T-PAKE in the password-only model. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014, Part II*, volume 8874 of *Lecture Notes in Computer Science*, pages 233–253, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014. Springer, Heidelberg, Germany.

[KMO90] Joe Kilian, Silvio Micali, and Rafail Ostrovsky. Minimum resource zero-knowledge proofs (extended abstract). In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 545–546, Santa Barbara, CA, USA, August 20–24, 1990. Springer, Heidelberg, Germany.

[KNYY19] Shuichi Katsumata, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Designated verifier/prover and preprocessing NIZKs from Diffie-Hellman assumptions. In *EUROCRYPT*, 2019.

[KW18] Sam Kim and David J. Wu. Multi-theorem preprocessing NIZKs from lattices. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 733–765, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany.

[LS91] Dror Lapidot and Adi Shamir. Publicly verifiable non-interactive zero-knowledge proofs. In Alfred J. Menezes and Scott A. Vanstone, editors, *Advances in Cryptology – CRYPTO'90*, volume 537 of *Lecture Notes in Computer Science*, pages 353–365, Santa Barbara, CA, USA, August 11–15, 1991. Springer, Heidelberg, Germany.

[Nao90] Moni Naor. Bit commitment using pseudo-randomness. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 128–136, Santa Barbara, CA, USA, August 20–24, 1990. Springer, Heidelberg, Germany.

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd Annual ACM Symposium on Theory of Computing*, pages 427–437, Baltimore, MD, USA, May 14–16, 1990. ACM Press.

[PsV06] Rafael Pass, abhi shelat, and Vinod Vaikuntanathan. Construction of a non-malleable encryption scheme from any semantically secure one. In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 271–289, Santa Barbara, CA, USA, August 20–24, 2006. Springer, Heidelberg, Germany.

[Rot10] Ron Rothblum. A taxonomy of enhanced trapdoor permutations. *Electronic Colloquium on Computational Complexity (ECCC)*, 17:145, 2010.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th Annual ACM Symposium on Theory of Computing*, pages 475–484, New York, NY, USA, May 31 – June 3, 2014. ACM Press.

A Hidden-Bits Generator from gap-CDH in Bilinear Groups

Let $(\mathbb{G}, \mathbb{G}_T, e, p, g) \leftarrow \mathsf{GroupGen}(\lambda)$ be a "bilinear group generator" which outputs cyclic groups $\mathbb{G}, \mathbb{G}_T$ of order p along with a generator g for $\mathbb{G}$ and a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$.

The gap-CDH assumption states that CDH holds in the group $\mathbb{G}$ even though the presence of a bilinear map makes DDH easy. In other words given g^x, g^y for random x, y, it should be hard to compute g^{xy} even though the bilinear map makes it easy to recognize the solution g^{xy} by checking $e(g^x, g^y) \stackrel{?}{=} e(g, g^{xy})$. Let hc be a hard-core bit associated with the CDH assumption in $\mathbb{G}$.

Let us define the following HBG:

- $\mathsf{Setup}(1^\lambda, 1^k)$: For all $i \in [k]$, pick a random $h_i \leftarrow \mathbb{G}$. Let γ be randomness for the hardcore bit hc. Output $\mathsf{crs} = (h_1, \ldots, h_k, \gamma)$.

- $\mathsf{GenBits}(\mathsf{crs})$: Pick a random $y \leftarrow \mathbb{Z}_p$, and compute $t_i = h_i^y$ for all $i \in [k]$. Output:

$$\begin{aligned}&\mathsf{com} = g^y,\\ &\{r_i = \mathsf{hc}(t_i;\gamma)\}_{i\in[k]}\\ &\{\pi_i = t_i\}_{i\in[k]}.\end{aligned}$$

- $\mathsf{Verify}(\mathsf{crs}, \mathsf{com}, i, r_i, \pi_i)$: Accept if $e(\mathsf{com}, h_i) = e(g, \pi_i)$ and $r_i = \mathsf{hc}(\pi_i;\gamma)$.

Theorem A.1. *The triple* $(\mathsf{Setup}, \mathsf{GenBits}, \mathsf{Verify})$ *is a weak Hidden-Bits Generator under the gap-CDH assumption.*

Proof. We prove completeness, succinctness, statistical binding and computational hiding.

Completeness: If $\mathsf{com} = g^y$ and $\pi_i = h_i^y$ and $r_i = \mathsf{hc}(h_i^y;\gamma)$ are correctly generated by $\mathsf{GenBits}$ then $e(\mathsf{com}, h_i) = e(g, h_i)^y = e(g, \pi_i)$ and $r_i = \mathsf{hc}(\pi_i;\gamma)$ so Verify accepts.

Succinctness: First, we have $|\mathcal{COM}| = |\mathbb{G}|$ is some fixed polynomial in the security parameter independent of k.

Statistical Binding: Define $\mathsf{Open}(1^k, \mathsf{crs}, \mathsf{com})$: compute y such that $g^y = \mathsf{com}$. Output:

$$\{r_i = \mathsf{hc}(h_i^y;\gamma)\}_{i\in[k]}\,.$$

If $\mathsf{Verify}(\mathsf{crs}, \mathsf{com}, i, r'_i, \pi_i)$ accepts then: firstly, $e(\mathsf{com}, h_i) = e(g, h_i)^y = e(g, \pi_i)$ which only occurs if $\pi_i = h_i^y$ and secondly $r'_i = \mathsf{hc}(\pi_i;\gamma) = \mathsf{hc}(h_i^y;\gamma)$. Therefore it must be the case that $r_i = r'_i$.

Computational Hiding: We want to prove that, for all $I \subseteq [k]$, given:

$$\left(\mathsf{crs} = (\{h_i\}_{i\in[k]}, \gamma), \mathsf{com} = g^y, \{r_i = \mathsf{hc}(h_i^y;\gamma)\}_{i\in I}, \{\pi_i = h_i^y\}_{i\in I}\right),$$

the values $\{r_j = \mathsf{hc}(h_j^y;\gamma)\}_{j\notin I}$ are indistinguishable from uniform. This follows via a hybrid argument over $j \notin I$. For each such j we first argue that, by the gap-CDH assumption, the value h_j^y is hard to compute even given all the other values – the reduction gets $h_j = g^x, g^y$, samples $x_i \leftarrow \mathbb{Z}_p$ and sets $h_i = g^{x_i}$ for all $i \neq j$ to simulate the adversary's input. Then we rely on the fact that hc is a hardcore bit to replace r_j by uniform.

□

B Hidden-Bits Generator from TDP

In this appendix we sketch the construction of a Hidden Bits Generator from doubly-enhanced trapdoor permutation. We follow notations of [GR13] and assume basic familiarity with the definitions of trapdoor permutations and their various enhancements as in [Gol11, Rot10, GR13, CL17]. For simplicity we assume that the trapdoor permutation has an efficiently recognizable index set and mention that the case that it does not can be handled via the techniques in [BY93, CL17].

Let $\{f_\alpha : D_\alpha \to D_\alpha\}_\alpha$ be a collection of doubly-enhanced trapdoor permutations. Let hc be an enhanced hardcore predicate for this collection (such a predicate exists without loss of generality by [GL89]). Using $\{f_\alpha\}_\alpha$ we construct a Hidden Bits Generator as follows. The CRS is $(s_1, \ldots, s_k)$ where each s_i is a random string for the domain sampling algorithm S of the TDP. Given these s_i's the $\mathsf{GenBits}$ algorithm selects a random index/trapdoor pair (α, τ) and sets α as the commitment. The hidden bits are now defined as $r_i = \mathsf{hc}(x_i)$, where $x_i = f_\alpha^{-1}(S(\alpha; s_i))$ (which can be computed using τ). The proofs string is simply $\{x_i\}_i$.

The Verify algorithm accepts, given α, s_i, x_i and r_i, if and only if α belongs to the index set (recall that we assumed that this set is efficiently recognizable) $r_i = \mathsf{hc}(x_i)$ and $f_\alpha(x_i) = S(\alpha; s_i)$.

Correctness is immediate. The fact that the commitment set is succinct follows from the fact that α was chosen independently of k. Statistical binding follows from the fact that since α specifies a permutation (and we are ensured of that since the verifier checks that it comes from the index set), then for every s_i the value of r_i is determined and there simply does not exist a preimage x_i of $S(\alpha; s_i)$ whose hardcore bit is not r_i.

The computational hiding property follows from the fact that hc is an enhanced hardcore predicate and a hybrid argument (where we used the "doubly-enhanced sampling algorithm" to make the hybrid argument go through).