New Constructions of Statistical NIZKs: Dual-Mode DV-NIZKs and More

Benoît Libert[1,2*], Alain Passelègue[2,3†], Hoeteck Wee[4‡], and David J. Wu[5§]

[1] CNRS, Laboratoire LIP, France
[2] ENS de Lyon, Laboratoire LIP (U. Lyon, CNRS, ENSL, Inria, UCBL), France
[3] Inria, France
[4] CNRS, ENS, PSL, Paris, France
[5] University of Virginia, Charlottesville, VA

Abstract. Non-interactive zero-knowledge proofs (NIZKs) are important primitives in cryptography. A major challenge since the early works on NIZKs has been to construct NIZKs with a *statistical* zero-knowledge guarantee against unbounded verifiers. In the common reference string (CRS) model, such "statistical NIZK arguments" are currently known from k-Lin in a pairing-group and from LWE. In the (reusable) designated-verifier model (DV-NIZK), where a trusted setup algorithm generates a reusable verification key for checking proofs, we also have a construction from DCR. If we relax our requirements to *computational* zero-knowledge, we additionally have NIZKs from factoring and CDH in a pairing group in the CRS model, and from nearly *all* assumptions that imply public-key encryption (e.g., CDH, LPN, LWE) in the designated-verifier model. Thus, there still remains a gap in our understanding of statistical NIZKs in both the CRS and the designated-verifier models.

In this work, we develop new techniques for constructing statistical NIZK arguments. First, we construct statistical DV-NIZK arguments from the k-Lin assumption in *pairing-free* groups, the QR assumption, and the DCR assumption. These are the first constructions in pairing-free groups and from QR that satisfy statistical zero-knowledge. All of our constructions are secure even if the verification key is chosen maliciously (i.e., they are "malicious-designated-verifier" NIZKs), and moreover, they satisfy a "dual-mode" property where the CRS can be sampled from two computationally indistinguishable distributions: one distribution yields *statistical DV-NIZK arguments* while the other yields *computational DV-NIZK proofs*. We then show how to adapt our k-Lin construction in a pairing group to obtain new *publicly-verifiable* statistical NIZK arguments from pairings with a *qualitatively weaker* assumption than existing constructions of pairing-based statistical NIZKs.

*Email: benoit.libert@ens-lyon.fr. Part of this research was supported by the French ANR ALAMBIC project (ANR-16-CE39-0006).

†Email: alain.passelegue@inria.fr.

‡Email: wee@di.ens.fr. Supported in part by ERC Project aSCEND (H2020 639554).

§Email: dwu4@virginia.edu. Part of this work was done while visiting ENS de Lyon. Supported by NSF CNS-1917414 and a University of Virginia SEAS Research Innovation Award.

Our constructions follow the classic paradigm of Feige, Lapidot, and Shamir (FLS). While the FLS framework has traditionally been used to construct computational (DV)-NIZK proofs, we newly show that the same framework can be leveraged to construct dual-mode (DV)-NIZKs.

1 Introduction

Non-interactive zero-knowledge (NIZK) proofs [BFM88, GMR89] allow a prover to send a single message to convince a verifier that a statement is true without revealing anything beyond this fact. Although such NIZKs cannot exist in the plain model, they can be realized in the common reference string (CRS) model, where a trusted party generates and publishes a common reference string accessible to the prover and the verifier. Shortly after the introduction of NIZKs, numerous constructions have been developed in the CRS model from many classes of cryptographic assumptions such as factoring [BFM88, DMP87, FLS90, BY92, FLS99, DDO+01, Gro10, Gol11, GR13, CL18], pairing-based assumptions [CHK03, GOS06], and lattice-based assumptions [CCH+19, PS19]. We can also construct NIZKs in the random oracle model [FS86].

A major open problem since the early works on non-interactive zero-knowledge has been to construct NIZKs with a *statistical* zero-knowledge guarantee against *computationally-unbounded* verifiers (i.e., "statistical NIZK arguments"). Here, we only have constructions from the k-Lin family of assumptions over pairing groups [GOS06, GOS12] and LWE [PS19] (or circular-secure FHE [CCH+19]). If we relax the model and consider (reusable) designated-verifier NIZKs (DV-NIZKs), where the trusted party that generates the CRS also generates a *secret* verification key that is used to verify proofs, then the recent work of Chase et al. [CDI+19] provides an instantiation of a statistical DV-NIZK from the DCR assumption. In contrast, if we are satisfied with computational zero-knowledge, then we can additionally construct publicly-verifiable NIZKs in the CRS model from QR [BFM88], factoring [FLS99], and the CDH assumption over a pairing group [CHK03]. In the designated-verifier model, a recent line of works [QRW19, CH19, KNYY19a, KNYY19b, LQR+19] has provided constructions of computational DV-NIZKs from essentially all cryptographic assumptions known to imply public-key encryption. These include assumptions like CDH in a pairing-free group and LPN. Thus, there is still a gap in our understanding of statistical NIZKs in the CRS model, and especially in the designated-verifier model. In this work, we develop new techniques for constructing statistical NIZKs in both the standard CRS model as well as the (reusable) designated-verifier model, which we review below.

Reusable designated-verifier NIZKs. A key focus in this work is the designated-verifier model [PsV06, DFN06], where a trusted party generates the CRS together with a *secret* verification key that is used to verify proofs. In this work, we focus exclusively on *reusable* (i.e., multi-theorem) security where soundness holds even against a prover who has oracle access to the verification algorithm. We also consider the stronger *malicious-designated-verifier* model (MDV-NIZKs) introduced by Quach et al. [QRW19], where a trusted party only samples a

common reference string,[1] and the verifier is allowed to choose its public and secret key-pair, which is used to generate and verify proofs, respectively. Here, we require that zero-knowledge should hold even if the verifier samples its public key maliciously. As discussed in [QRW19], MDV-NIZKs are equivalent to 2-round zero-knowledge protocols in the CRS model where the verifier's initial message is reusable. A recent line of works have shown how to construct (M)DV-NIZKs with *computational* zero-knowledge from nearly all assumptions known to imply public-key encryption (e.g., CDH, LWE, LPN) [QRW19, CH19, KNYY19a, KNYY19b, LQR+19].

Several recent works have also explored other relaxations of the standard notion of publicly-verifiable NIZKs such as the reusable designated-prover model (where there is a secret proving key and a public verification key) [KW18, KNYY19a] or the reusable preprocessing model (where both the proving and verifications keys are secret) [BCGI18, BCG+19]. In this work, our focus is on reusable designated-verifier NIZKs and publicly-verifiable NIZKs.

Dual-mode NIZKs. An appealing feature of several existing NIZK constructions [GOS06, GOS12, PS19] is they satisfy a "dual-mode" property. Namely, the CRS in these schemes can be sampled from one of two computationally indistinguishable distributions. One distribution yields *computational NIZK proofs* while the other yields *statistical NIZK arguments*. Dual-mode NIZKs are powerful primitives and a recent work has also studied generic constructions from obfuscation [HU19]. Most of the constructions we develop in this work naturally satisfy this dual-mode property.

1.1 Our Results

In this work, we develop new techniques for constructing statistical NIZKs for general NP languages that yield new constructions in both the reusable designated-verifier model and the standard CRS model. Our techniques enable the following new constructions:

- Under the k-Lin assumption in a *pairing-free* group (for any $k \geq 1$; recall that 1-Lin $\equiv$ DDH), we obtain a statistical MDV-NIZK argument in the common *random* string model and a computational MDV-NIZK proof in the common *reference* string model.[2] This is the first construction of a statistical DV-NIZK argument (even ignoring malicious security) in a pairing-free group, and the first construction of a computational MDV-NIZK proof from a *static* assumption. Previously, computational MDV-NIZK proofs were only known from the interactive "one-more CDH" assumption [QRW19].

[1] In [QRW19], they require the stronger notion where the CRS is a *uniformly random string*. In some of our constructions in this work, the CRS will be a *structured* string. We believe that this model is still meaningful as the CRS just needs to be sampled once and can be reused by arbitrarily many verifiers, and zero-knowledge holds as long as the CRS is properly sampled.

[2] This is in fact a dual-mode NIZK, where one of the CRS distributions corresponds to the *uniform* distribution.

- Under the k-Lin assumption in $\mathbb{G}_1$ and the k-KerLin assumption in $\mathbb{G}_2$ of a pairing group (for any $k \geq 1$), we obtain a *publicly-verifiable* statistical NIZK argument in the common reference string model. Notably, the k-KerLin assumption is a *search* assumption that is implied by the standard k-Lin assumption [MRV15, KW15]. This is a *qualitatively weaker* assumption than existing pairing-based constructions of statistical NIZK arguments which rely on a *decisional* assumption (k-Lin) in *both* $\mathbb{G}_1$ and $\mathbb{G}_2$ [GOS06, GOS12].
- Under the QR assumption, we obtain a dual-mode MDV-NIZK in the common reference string model. Previously, we could only construct (publicly-verifiable) *computational* NIZKs from the QR assumption [BFM88] (or more generally, from factoring [FLS90, FLS99]), but nothing was known for statistical NIZKs or DV-NIZKs from these assumptions.
- Under the DCR assumption, we obtain a dual-mode MDV-NIZK in the common reference string model. This matches the recent construction described in [CDI+19], which realizes the result through a different approach (via reusable non-interactive secure computation).

We provide a detailed comparison of our constructions with existing NIZK constructions (in both the designated-verifier and the publicly-verifiable models) in Table 1. We describe the formal instantiations in Section 5.

From FLS to statistical NIZKs. All of our constructions follow the classic paradigm of Feige, Lapidot, and Shamir (FLS) [FLS99] who provide a general compiler from a NIZK in an idealized model (i.e., the "hidden-bits" model) to a computational NIZK proof in the CRS model. To date, all existing instantiations of the [FLS99] paradigm have yielded *computational NIZK proofs* in either the CRS model [FLS90, BY92, FLS99, CHK03, Gro10, Gol11, GR13, CL18] or the designated-verifier model [QRW19, CH19, KNYY19a]. In this work, we show how to adapt the general FLS paradigm to obtain new constructions of *statistical NIZK arguments* and more generally, *dual-mode NIZKs*. We provide a general overview of our techniques in Section 1.2.

We further note that previous statistical NIZK arguments from pairings, LWE, and DCR follow very different approaches. Our work can also be viewed as providing a *unified* approach to realizing these existing results—both computational and statistical, with the sole exception of the LWE-based scheme—via the FLS paradigm, while also improving upon some of these prior results, and obtaining new ones.

1.2 Technical Overview

We begin with a brief overview of the Feige-Lapidot-Shamir (FLS) framework [FLS90, FLS99] for constructing NIZK proofs for NP. We then describe how to adapt the main ideas from the FLS framework to obtain new constructions of (malicious) designated-verifier dual-mode NIZKs as well as publicly-verifiable statistical NIZK arguments.

Construction	Model	Soundness	ZK	Assumption
[BFM88]	public	stat.	comp.	QR
[FLS90, FLS99]	public	stat.	comp.	trapdoor permutation
[SW14]	public	comp.	perf.	$i\mathcal{O}$ + one-way function
[CHK03]*	public	stat.	comp.	CDH ($\mathbb{G}_2$)
[GOS06, GOS12]*	public	perf./comp.	comp./perf.	k-Lin ($\mathbb{G}_1, \mathbb{G}_2$)
This work*	**public**	**comp.**	**stat.**	k-Lin ($\mathbb{G}_1$), k-KerLin ($\mathbb{G}_2$)†
[PS19]	public	stat./comp.	comp./stat.	LWE
[QRW19, CH19, KNYY19a]	DV	stat.	comp.	CDH
[QRW19]	MDV	stat.	comp.	one-more CDH
[LQR+19]	MDV	comp.	comp.	CDH/LWE/LPN
[CDI+19]	MDV	stat./comp.	comp./stat.	DCR
This work	**MDV**	**stat./comp.**	**comp./stat.**	k-Lin‡/QR/DCR

*This is a *pairing-based* construction. In the assumption column, we enumerate all of the necessary hardness assumptions to instantiate the scheme (in an asymmetric setting).
†The k-KerLin refers to the kernel k-Lin assumption [MRV15, KW15], which can be viewed as the *search* analog of the classic k-Lin assumption [BBS04, HK07, Sha07].
‡This is over a *pairing-free* group. The special case where $k = 1$ corresponds to the standard DDH assumption. In addition, if we consider the vanilla DV-NIZK model (without malicious security), there is a simple instantiation (over elliptic-curve groups) that achieves *perfect* zero-knowledge.

Table 1: Comparison of our construction to existing multi-theorem NIZKs. We write "public" to denote the standard CRS model (with public proving and public verification), "DV" to denote the designated-verifier model, and "MDV" to denote the malicious-designated-verifier model. For soundness and zero-knowledge, we write "comp." to denote the computational variant of the property, "stat." to denote the statistical variant, and "perf." to denote the perfect variant. When a scheme supports a dual-mode CRS, we indicate the two modes by writing "stat./comp." For the pairing-based constructions, we list the necessary assumptions needed within each of the base groups $\mathbb{G}_1$ and $\mathbb{G}_2$ (assuming an asymmetric pairing).

The FLS framework. The starting point of the FLS construction is a NIZK in an idealized model called the "hidden-bits model." In this model, a trusted party generates a string of uniformly random bits $r_1, \ldots, r_\rho \in \{0,1\}$ and gives them to the prover. The prover then outputs a proof π along with a set of indices $I \subseteq [\rho]$. The verifier receives $(\pi, \{r_i\}_{i \in I})$ from the trusted party. The model guarantees that the prover cannot influence the value of any of the r_i's and the verifier does not learn anything about r_i for indices $i \notin I$. Feige et al. [FLS99] showed how to construct a NIZK with statistical soundness and perfect zero-knowledge in the hidden-bits model by adapting Blum's Σ-protocol for graph Hamiltonicity [Blu86]. Next, the FLS construction compiles a NIZK in the hidden-bits model into one in the CRS model by using the CRS to define the sequence of hidden bits. We recall the FLS compiler based on trapdoor permutations:

– The CRS contains the description of a *family* of trapdoor permutations over $\{0,1\}^\lambda$ together with ρ random strings $w_1, \ldots, w_\rho \in \{0,1\}^\lambda$ that are used to define a string of ρ hidden bits.

– A hidden-bits string is defined by sampling a permutation σ from the family of trapdoor permutations specified by the CRS, along with a trapdoor for computing σ^{-1}. In conjunction with w_i in the CRS, the permutation σ defines a hidden bit $r_i := \mathsf{hc}(\sigma^{-1}(w_i))$, where $\mathsf{hc}(\cdot)$ is a hard-core bit of σ. We refer to σ as a "commitment" to the hidden-bits string $r \in \{0,1\}^\rho$.
– The prover can open a commitment σ to a bit r_i by sending (i, r_i, u_i) where $u_i := \sigma^{-1}(w_i)$. The verifier checks that $\sigma(u_i) = w_i$ and that $\mathsf{hc}(u_i) = r_i$.

The security argument proceeds roughly as follows:

– Since hc is a hard-core bit, the value of any unopened bit r_i is *computationally hidden* given σ and w_i. The resulting NIZK satisfies computational zero-knowledge.
– The permutation σ and the string w_i *statistically determine* r_i, and the prover cannot open r_i to any value other than $\mathsf{hc}(\sigma^{-1}(w_i))$. The resulting NIZK satisfies statistical soundness. Note that a cheating prover can bias the bit r_i due to the adaptive choice of σ. The FLS construction works around this by leveraging the fact that if the commitment σ has length ℓ, then a malicious prover can bias at most ℓ of the ρ bits, and soundness holds as long as $\ell \ll \rho$.

Our approach. In this work, we start by showing how to realize a *dual-mode* variant of the hidden-bits model in the *designated-verifier* setting where the underlying commitment to the random bits is either statistically binding or statistically hiding. This "dual-mode" property yields either a *computational DV-NIZK proof* or a *statistical DV-NIZK argument* depending on how the CRS is sampled (similar to previous dual-mode NIZKs [GOS06, GOS12, PS19]). We then show how to extend one of our constructions to the *publicly-verifiable* setting.

An instantiation from DDH. We first sketch our construction from the DDH assumption. Here, we will work with a (multiplicative) group $\mathbb{G}$ of prime order p and generator g. For a vector $\mathbf{v} = (v_1, \ldots, v_n) \in \mathbb{Z}_p^n$, we write $g^{\mathbf{v}}$ to denote a vector of group elements $(g^{v_1}, \ldots, g^{v_n})$. Analogous to the FLS construction from trapdoor permutations, the CRS contains

– the description $g^{\mathbf{v}}$ of a function, where $\mathbf{v} \xleftarrow{\mathrm{R}} \mathbb{Z}_p^{\rho+1}$ and $g^{\mathbf{v}}$ plays a role similar to the *family* of trapdoor permutations in the FLS construction;
– $g^{\mathbf{w}_1}, \ldots, g^{\mathbf{w}_\rho}$ where each $\mathbf{w}_i \in \mathbb{Z}_p^{\rho+1}$ plays a role similar to $w_i \in \{0,1\}^\lambda$.

In our construction, we will vary the distribution of $\mathbf{w}_i$ (but *not* $\mathbf{v}$) as follows:

– If we want *statistically-binding* "hidden bits," then we sample $\mathbf{w}_i \leftarrow s_i \mathbf{v}$, where $s_i \xleftarrow{\mathrm{R}} \mathbb{Z}_p$.
– If we want *statistically-hiding* "hidden bits," then we sample $\mathbf{w}_i \xleftarrow{\mathrm{R}} \mathbb{Z}_p^{\rho+1}$.

Thanks to the DDH assumption, $(g^{\mathbf{v}}, g^{s_i \mathbf{v}})$ is pseudorandom, and therefore, these two CRS distributions are computationally indistinguishable.[3] As with

[3]This idea of encoding either a full-rank matrix in the exponent or a rank-1 matrix in the exponent also featured in the construction of lossy public-key encryption from the Matrix Diffie-Hellman assumptions [HJR16].

the construction from trapdoor permutations, the hidden bit r_i is a function of the CRS components $g^{\mathbf{v}}, g^{\mathbf{w}_i}$ together with an additional message σ from the prover. Concretely, the prover samples a random $\mathbf{y} \xleftarrow{R} \mathbb{Z}_p^{\rho+1}$ and sends $\sigma = g^{\mathbf{y}^\mathsf{T}\mathbf{v}} \in \mathbb{G}$. In conjunction with $g^{\mathbf{w}_i}$ in the CRS, the vector $\mathbf{y}$ defines a hidden bit $r_i := H(g^{\mathbf{y}^\mathsf{T}\mathbf{w}_i})$, where $H\colon \mathbb{G} \to \{0,1\}$ is a universal hash function. Importantly, while the description $g^{\mathbf{v}}, g^{\mathbf{w}_1}, \ldots, g^{\mathbf{w}_\rho}$ in the CRS grows with ρ, the prover's message σ does not. Now, observe that:

- In binding mode where $\mathbf{w}_i = s_i\mathbf{v}$, we have $\mathbf{y}^\mathsf{T}\mathbf{w}_i = s_i\mathbf{y}^\mathsf{T}\mathbf{v}$. Then, $r_i = H(g^{\mathbf{y}^\mathsf{T}\mathbf{w}_i}) = H(g^{s_i\mathbf{y}^\mathsf{T}\mathbf{v}}) = H(\sigma^{s_i})$ is fully determined by the commitment $\sigma = g^{\mathbf{y}^\mathsf{T}\mathbf{v}}$ together with $g^{\mathbf{v}}, g^{\mathbf{w}_i}$ in the CRS.
- In hiding mode where $\mathbf{w}_i \xleftarrow{R} \mathbb{Z}_p^{\rho+1}$, the quantity $g^{\mathbf{y}^\mathsf{T}\mathbf{w}_i}$ is completely hidden given $g^{\mathbf{y}^\mathsf{T}\mathbf{v}}$ along with $g^{\mathbf{v}}, g^{\mathbf{w}_i}$ in the CRS, provided that $\mathbf{v}$ and $\mathbf{w}_i$ are linearly independent. More generally, perfect hiding holds as long as the vectors $\mathbf{v}, \mathbf{w}_1, \ldots, \mathbf{w}_\rho$ are linearly independent over $\mathbb{Z}_p^{\rho+1}$.

Next, to open the bit r_i, the prover will send along $g^{\mathbf{y}^\mathsf{T}\mathbf{w}_i}$. To ensure that a cheating prover computes this quantity correctly in the *designated-verifier* model, we rely on techniques using the Cramer-Shoup hash-proof system [CS98, CS02, CKS08] (and also used to construct computational DV-NIZK proofs from CDH [QRW19, CH19, KNYY19a]):

- The verifier's public key consists of components $g^{\mathbf{z}_i} := g^{a\mathbf{w}_i + b_i\mathbf{v}}$ where $a, b_i \xleftarrow{R} \mathbb{Z}_p$ are secret coefficients chosen by the verifier. The *secret* verification key is the scalars $(a, b_1, \ldots, b_\rho)$.
- The prover sends $g^{u_i} := g^{\mathbf{y}^\mathsf{T}\mathbf{z}_i} \in \mathbb{G}$ in addition to $\sigma = g^c := g^{\mathbf{y}^\mathsf{T}\mathbf{v}} \in \mathbb{G}$ and $g^{t_i} := g^{\mathbf{y}^\mathsf{T}\mathbf{w}_i} \in \mathbb{G}$.
- The verifier checks that $g^{u_i} = (g^{t_i})^a (g^c)^{b_i}$ using (a, b_i).

In the statistically-binding mode where $\mathbf{w}_i = s_i\mathbf{v}$, we have $\mathbf{z}_i = (as_i + b_i)\mathbf{v}$, so (a, b_i) has (statistical) entropy given $\mathbf{v}, \mathbf{w}_i, \mathbf{z}_i$. Roughly speaking, reusable soundness then follows from the analysis of the Cramer-Shoup CCA-secure encryption scheme [CS98, CS02, CKS08] to enforce the consistency check $t_i = s_i c$. In conjunction with a NIZK in the hidden-bits model, we thus obtain a dual-mode DV-NIZK from the DDH assumption. This construction generalizes very naturally to the k-Lin family of assumptions [BBS04, HK07, Sha07, EHK+13] for any $k \geq 1$ (where in particular, 1-Lin is the DDH assumption). Concretely, we make the following substitutions to the above construction:

$$\begin{aligned} \mathbf{v} \in \mathbb{Z}_p^{\rho+1} &\mapsto \mathbf{V} \in \mathbb{Z}_p^{(\rho+1)\times k} \\ s_i, b_i \in \mathbb{Z}_p &\mapsto \mathbf{s}_i, \mathbf{b}_i \in \mathbb{Z}_p^k \\ t_i, u_i, c \in \mathbb{Z}_p &\mapsto \mathbf{t}_i, \mathbf{u}_i, \mathbf{c} \in \mathbb{Z}_p^k \end{aligned}$$

We provide the full details and security analysis in the full version.

Extending to QR/DCR. Our DDH construction readily generalizes to the subgroup indistinguishability family of assumptions [BG10] (which generalize the QR [GM82] and DCR [Pai99] assumptions). While there are some technical differences in our concrete instantiations from QR and DCR, all of the main ideas can be described via the conceptually-simpler language of subgroup indistinguishability. This is the approach we take in this overview, and we refer to the technical sections for the full details. First, the subgroup indistinguishability assumption says that the distributions (g, h, g^{r_1}) and $(g, h, g^{r_1} h^{r_2})$ are computationally indistinguishable, where g, h generate subgroups of co-prime order m_g, m_h, respectively, and $r_1 \xleftarrow{R} \mathbb{Z}_{m_g}$, $r_2 \xleftarrow{R} \mathbb{Z}_{m_h}$.

Similar to the DDH instantiation, the CRS contains a function $g^{\mathbf{v}}$ (where $\mathbf{v} \xleftarrow{R} \mathbb{Z}^{\rho}_{m_g m_h}$) together with additional components $g^{s_1 \mathbf{v}} h^{\hat{\mathbf{w}}_1}, \ldots, g^{s_\rho \mathbf{v}} h^{\hat{\mathbf{w}}_\rho}$, where $\hat{\mathbf{w}}_i = \mathbf{0}$ in binding mode and $\hat{\mathbf{w}}_i = \mathbf{e}_i$ in hiding mode. Here $\mathbf{e}_i$ is the basis vector whose i^{th} index is 1. Under the subgroup indistinguishability assumption, these two distributions are computationally indistinguishable.

Next, the hidden bit r_i is a function of the CRS components $g^{\mathbf{v}}$ and $g^{s_i \mathbf{v}} h^{\hat{\mathbf{w}}_i}$ together with an additional commitment σ from the prover. Specifically, the prover samples a vector $\mathbf{y} = (y_1, \ldots, y_\rho) \xleftarrow{R} \mathbb{Z}^{\rho}_{m_g m_h}$ and computes

$$\sigma := g^{\mathbf{y}^\top \mathbf{v}} \quad \text{and} \quad t_i := g^{s_i \mathbf{y}^\top \mathbf{v}} h^{\mathbf{y}^\top \hat{\mathbf{w}}_i} \quad \text{and} \quad r_i := H(t_i), \tag{1.1}$$

where H is a hash function. Now, observe that:

- In binding mode where $\hat{\mathbf{w}}_i = \mathbf{0}$, then $t_i = g^{s_i \mathbf{y}^\top \mathbf{v}} = \sigma^{s_i}$. Thus, t_i (and correspondingly, r_i) is fully determined by the commitment σ and the components $g^{\mathbf{v}}$, $g^{s_i \mathbf{v}} h^{\hat{\mathbf{w}}_i} = g^{s_i \mathbf{v}}$ in the CRS.
- In hiding mode where $\hat{\mathbf{w}}_i = \mathbf{e}_i$, then $t_i = g^{s_i \mathbf{w}^\top \mathbf{y}} h^{y_i}$. Since g and h generate subgroups of co-prime order m_g and m_h, respectively, we can appeal to the Chinese remainder theorem to argue that the commitment $\sigma = g^{\mathbf{y}^\top \mathbf{v}}$ *perfectly* hides the value of $\mathbf{y} \bmod m_h$. Since $\mathbf{y}$ is uniform over $\mathbb{Z}_{m_g m_h}$, this means that $t_1, \ldots, t_i$ have at least $\log m_h$ bits of statistical entropy given σ (and the components of the CRS).

  In the DCR construction, $m_h = N$ is a product of two large primes, so we can use a standard universal hash function to extract a uniformly random bit [HILL99].

  In the QR construction, $m_h = 2$, so each component t_i contains just one bit of entropy, and we cannot appeal to the leftover hash lemma. In this case, we adapt an idea from [DGI+19] (for constructing trapdoor hash functions from QR) and use a deterministic function to extract the bit from t_i.

Finally, to open a bit r_i, the prover provides σ, t_i, along with a proof that t_i and σ are consistent (i.e., there exists some $\mathbf{y}$ such that Eq. (1.1) hold). Here, we use the same techniques as in the DDH setting (i.e., using the Cramer-Shoup hash-proof system) to implement this. In the QR setting, we encounter some challenges because the order of the subgroup generated by h is polynomial-sized,

which allows the adversary to break soundness with noticeable probability. To amplify soundness, we essentially embed multiple copies of the Cramer-Shoup hash-proof system and ensure that the proof verifies only if *all* copies verify (while retaining *reusable* soundness). We refer to the full version for the full analysis of the QR and DCR constructions.

Handling malicious verifiers. All of the constructions described thus far are zero-knowledge only if the verifier samples its public verification key *honestly*. However, if the verifier can choose its key *arbitrarily*, then it can break zero-knowledge. To see this, consider again the DDH construction (in hiding mode). There, the CRS contains elements $g^{\mathbf{v}}, g^{\mathbf{w}_1}, \ldots, g^{\mathbf{w}_\rho}$, and a verifier's public key is $(g^{\mathbf{z}_1}, \ldots, g^{\mathbf{z}_\rho})$ where $\mathbf{z}_i = a\mathbf{w}_i + b_i\mathbf{v}$. To generate a hidden-bits string r, the prover samples $\mathbf{y} \xleftarrow{R} \mathbb{Z}_p^{\rho+1}$ and sets $r_i = H(g^{\mathbf{y}^\top \mathbf{w}_i})$. To open a bit r_i, the prover computes $g^{t_i} = g^{\mathbf{y}^\top \mathbf{w}_i}$ and $g^{u_i} = g^{\mathbf{y}^\top \mathbf{z}_i}$. In order to appeal to security of the underlying NIZK in the hidden-bits model, we require that the commitment $\sigma = g^{\mathbf{y}^\top \mathbf{v}}$, the value of r_i, and the opening (g^{t_i}, g^{u_i}) do not leak information about any other (unopened) bit r_j. This is the case when all of the verification key components $\mathbf{z}_i$ are generated honestly. In this case, $\mathbf{v}, \mathbf{w}_1, \ldots, \mathbf{w}_\rho$ are linearly independent, and $\mathbf{z}_i$ is a function of only $\mathbf{v}$ and $\mathbf{w}_i$. However, a malicious verifier can choose $\mathbf{z}_i = \mathbf{w}_j$ for some $j \neq i$. Then, if the honest prover computes an opening to r_i, it will also compute $g^{u_i} = g^{\mathbf{y}^\top \mathbf{z}_i} = g^{\mathbf{y}^\top \mathbf{w}_j}$, which completely leaks the value of r_j. As such, the basic scheme is insecure against a malicious verifier.

This problem where an opening to r_i can leak information about the value r_j for $j \neq i$ is the same problem encountered in the basic DV-NIZK from [QRW19]. In this work, we adopt the same general strategy as them to defend against malicious verifiers. At a high-level, the approach of [QRW19] for achieving security against malicious verifiers is to use the basic scheme above to generate a hidden-bits string $r'_1, \ldots, r'_\ell$ of length $\ell \gg \rho$. Each of the ρ hidden bits $r_1, \ldots, r_\rho$ is then derived as a sparse pseudorandom combination of the bits $r'_1, \ldots, r'_\ell$. More specifically, the prover chooses a mapping φ that maps each index $i \in [\rho]$ onto a set $\varphi(i) \subseteq [\ell]$. Each bit r_i is a deterministic function of r'_j for $j \in \varphi(i)$. To open a bit r_i, the prover instead opens up all bits r'_j for $j \in \varphi(i)$. The length ℓ and the size $|\varphi(i)|$ of the sets are chosen so as to ensure that for all unopened bits $j \in [\rho]$, there is at least one index $k \in \varphi(j)$ such that r'_k is hidden from the verifier, which ideally, is sufficient to mask the value of r_j. Quach et al. show how to implement this idea by relying on a one-more CDH assumption (in conjunction with somewhere equivocal PRFs [HJO+16]), and a complex rewinding argument in the security proof. In our setting, the algebraic structure of our construction enables us to make a conceptually-simpler *information-theoretic* argument (and only needing to assume a PRG). As such, we are able to obtain a *dual-mode* MDV-NIZK from the DDH (and more generally, k-Lin), QR, and DCR assumptions.

We give a brief overview of how we extend the basic DDH construction sketched above to achieve security against malicious verifiers. The same idea extends to the QR and DCR constructions. Specifically, we use our basic construction to generate a hidden-bits string of length $\ell \gg \rho$ as follows:

– The CRS (in hiding mode) consists of group elements $g^{\mathbf{v}}, g^{\mathbf{w}_1}, \ldots, g^{\mathbf{w}_\ell}$, where $\mathbf{v}, \mathbf{w}_1, \ldots, \mathbf{w}_\ell \xleftarrow{R} \mathbb{Z}_p^{\ell+1}$. With overwhelming probability, these vectors are linearly independent.
– The honest verifier's public key is $(g^{\mathbf{z}_1}, \ldots, g^{\mathbf{z}_\ell})$, constructed in the usual manner.
– The prover's commitment is a vector $\mathbf{y} \in \mathbb{Z}_p^{\ell+1}$ as well as a seed s for a PRG.[4] The PRG outputs a collection of ρ blocks, where each block consists of a set $S_i \subseteq [\ell]$ and a vector $\boldsymbol{\alpha} \in \mathbb{Z}_p^{\ell}$. The hidden bit r_i is determined by first computing $g^{t_j} = g^{\mathbf{y}^\mathsf{T}\mathbf{w}_j}$ for all $j \in S_i$ and defining $r_i := H(\prod_{j \in S_i} g^{\alpha_j t_j})$.
– The opening for r_i consists of $g^{t_j} = g^{\mathbf{y}^\mathsf{T}\mathbf{w}_j}$ and $g^{u_j} = g^{\mathbf{y}^\mathsf{T}\mathbf{z}_j}$ for all $j \in S_i$.

Our goal is to show that even for an adversarially-chosen verification key, the commitment σ and the opening $(\{g^{t_j}, g^{u_j}\}_{j \in S_i})$ to a bit r_i does not leak any information about r_j whenever $j \neq i$.[5] By construction, the opening to r_i is determined by $\mathbf{y}^\mathsf{T}\mathbf{v}$, $\mathbf{y}^\mathsf{T}\mathbf{w}_j$, and $\mathbf{y}^\mathsf{T}\mathbf{z}_j$ for $j \in S_i$ (where the set S_i is *pseudorandom*). Take any index $i^* \neq i$. Then, if there exists $j^* \in \varphi(i^*)$ such that $\mathbf{w}_{j^*}$ is linearly independent of $\{\mathbf{v}, \mathbf{w}_j, \mathbf{z}_j\}_{j \in S_i}$, then the value of $\mathbf{y}^\mathsf{T}\mathbf{w}_{j^*}$ is independent and uniformly random given the view of the adversary (since the honest prover samples $\mathbf{y} \xleftarrow{R} \mathbb{Z}_p^{\ell+1}$). In this case, the value $g^{t_{j^*}} = g^{\mathbf{y}^\mathsf{T}\mathbf{w}_{j^*}}$ remains uniformly random and statistically hides r_{i^*}. Thus, it suffices to set ℓ and $|S_i|$ so that there will always exist $j^* \in \varphi(i^*)$ where $\mathbf{w}_{j^*}$ is linearly independent of $\{\mathbf{v}, \mathbf{w}_j, \mathbf{z}_j\}_{j \in S_i}$ with overwhelming probability. In the case of our DDH construction, we can set $|S_i| = \lambda$, where λ is a security parameter, and $\ell = 3\rho^2\lambda$ to satisfy this property. We provide the details of our DDH (more generally, its generalization to the k-Lin assumption) in Section 4.3 and our QR and DCR constructions in the full version.

Public verifiability via pairings. All of the constructions we have described so far operate in the designated-verifier model because our constructions rely on a Cramer-Shoup-style hash proof system to argue consistency between a commitment and the opening. If we can instead *publicly* check consistency between a commitment and its opening, then the resulting scheme becomes publicly verifiable. For the DDH construction, we can implement the consistency check using a pairing (this is the approach taken in [CHK03] to obtain a computational NIZK proof). In this work, we develop a similar approach to obtain a statistical NIZK argument from pairings.

In particular, let $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be an (asymmetric) pairing. Let g_1, g_2 be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. At a high level, we implement the

[4] We require a PRG because the prover's message needs to be *succinct* in order to argue soundness of the resulting NIZK in the FLS paradigm. Thus, we rely on a PRG for compression. Note that even though we rely on a computational assumption, we can still show *statistical* zero-knowledge. The security proof only requires that there are no efficient statistical tests that can distinguish the output of the PRG from a random string (which is implied by PRG security).

[5] To show adaptive, multi-theorem zero-knowledge, we in fact show an even stronger *simulation* property. We refer to Section 3 for more details.

DDH scheme in $\mathbb{G}_1$ and use $\mathbb{G}_2$ for verification. More specifically, the CRS is $g_1^{\mathbf{y}}, g_1^{\mathbf{w}_1}, \ldots, g_1^{\mathbf{w}_\rho}$, and the verification key is $g_1^{(a\mathbf{w}_1+b_1\mathbf{v})}, \ldots, g_1^{(a\mathbf{w}_\rho+b_\rho\mathbf{v})}$. The commitment, hidden-bits sequence, and openings are defined as before:

$$\sigma = g_1^c = g_1^{\mathbf{y}^\mathsf{T}\mathbf{v}} \quad , \quad r_i = H(g_1^{\mathbf{y}^\mathsf{T}\mathbf{w}_i}) \quad , \quad g_1^{t_i} = g_1^{\mathbf{y}^\mathsf{T}\mathbf{w}_i} \quad \text{and} \quad g_1^{u_i} = g_1^{\mathbf{y}^\mathsf{T}(a\mathbf{w}_i+b_i\mathbf{v})}.$$

In the designated-verifier setting, the verifier checks $g_1^{u_i} \stackrel{?}{=} (g_1^{t_i})^a (g_1^c)^{b_i}$. A direct approach for public verification is to include $g_2^a, g_2^{b_1}, \ldots, g_2^{b_\rho}$ as part of the verification key, and check the following:

$$e(g_1^{u_i}, g_2) \stackrel{?}{=} e(g_1^{t_i}, g_2^a) \cdot e(g_1^c, g_2^{b_i}).$$

While this approach is *correct*, it is unclear to argue soundness (even against computationally-bounded adversaries). In the designated-verifier setting, the soundness analysis critically relies on the verification coefficients a, b_i being hidden from the adversary, and it is unclear how to make such an argument when the adversary is given $g_2^a, g_2^{b_i}$.

To base hardness on a concrete cryptographic assumption, we leverage a technique from [KW15], who describe a general method to "securely publish" the verification key in the exponent (as we hoped to do in our initial attempt above) with a concrete security reduction to a *search* assumption in $\mathbb{G}_2$. This yields a general compiler from a designated-verifier scheme with unconditional soundness to a publicly-verifiable scheme with computational soundness, at the expense of requiring a pairing and a *search* assumption in $\mathbb{G}_2$. The compiler preserves zero-knowledge of the underlying scheme.

Concretely, instead of scalar verification coefficients a, b_i, we instead sample vectors $\mathbf{a}, \mathbf{b}_i \xleftarrow{\mathrm{R}} \mathbb{Z}_p^2$, and publish $g_1^{\mathbf{w}_i\mathbf{a}^\mathsf{T}+\mathbf{v}\mathbf{b}_i^\mathsf{T}}$ for each $i \in [\rho]$ in the CRS. The public verification components will consist of $g_2^{\mathbf{d}}, g_2^{\mathbf{a}^\mathsf{T}\mathbf{d}}, g_2^{\mathbf{b}_1^\mathsf{T}\mathbf{d}}, \ldots, g_2^{\mathbf{b}_\rho^\mathsf{T}\mathbf{d}}$, where $\mathbf{d} \in \mathbb{Z}_p^2$. The key observation is that $\mathbf{a}, \mathbf{b}_1, \ldots, \mathbf{b}_\rho$ have *statistical entropy* even given the public components $g_2^{\mathbf{d}}, g_2^{\mathbf{a}^\mathsf{T}\mathbf{d}}, g_2^{\mathbf{b}_1^\mathsf{T}\mathbf{d}}, \ldots, g_2^{\mathbf{b}_\rho^\mathsf{T}\mathbf{d}}$. The commitment, hidden-bits sequence, and openings are still computed as before, except the verification component $g_1^{u_i}$ is replaced with $g_1^{\mathbf{u}_i^\mathsf{T}} = g_1^{\mathbf{y}^\mathsf{T}(\mathbf{w}_i\mathbf{a}^\mathsf{T}+\mathbf{v}\mathbf{b}_i^\mathsf{T})}$. The verification relation now checks

$$e(g_1^{\mathbf{u}_i^\mathsf{T}}, g_2^{\mathbf{d}}) \stackrel{?}{=} e(g_1^{t_i}, g_2^{\mathbf{a}^\mathsf{T}\mathbf{d}}) \cdot e(g_1^c, g_2^{\mathbf{b}_i^\mathsf{T}\mathbf{d}}).$$

Since the verification coefficients $\mathbf{a}, \mathbf{b}_1, \ldots, \mathbf{b}_\rho$ have statistical entropy given the public key, we can appeal to DDH in $\mathbb{G}_1$ and the 1-KerLin assumption (a *search* assumption that is *weaker* than DDH) over $\mathbb{G}_2$ to argue soundness of the resulting construction. This yields a publicly-verifiable statistical NIZK argument in the common *reference* string model. We provide the full description and analysis (generalized to the k-Lin and k-KerLin family of assumptions for any $k \geq 1$) in the full version.

Our pairing-based construction does not appear to have a dual mode and it is unclear how to modify this construction to obtain computational NIZK proofs.

We do note that computational NIZK proofs can be built directly from pairings (under the CDH assumption in $\mathbb{G}_1$) also by following the FLS paradigm [CHK03]. At the same time, it is also unclear how to adapt the [CHK03] construction to obtain statistical NIZK arguments.

A unifying abstraction: dual-mode hidden-bits generators. We unify the different algebraic constructions through the abstraction of a "dual-mode hidden-bits generator." Previously, Quach et al. [QRW19] introduced the notion of a *hidden-bits generator* (HBG) and showed how to use an HBG to implement the classic FLS paradigm in both the designated-verifier and the publicly-verifiable settings. Very briefly, an HBG with output size ρ consists of four main algorithms (Setup, KeyGen, GenBits, Verify):

- The Setup algorithm outputs a common reference string crs, and KeyGen generates a public key pk along with a (possibly secret) verification key sk.
- The GenBits algorithm outputs a short commitment σ together with a sequence of hidden bits $r \in \{0,1\}^\rho$ as well as openings $\{\pi_i\}_{i\in[\rho]}$.
- The Verify algorithm takes an index $i \in [\rho]$, a bit $r_i \in \{0,1\}$, and an opening π_i and either accepts or rejects the proof.

The main security requirements are *statistical binding* (i.e., no adversary can produce a commitment σ and valid openings π_i, π_i' that open to 0 and 1 for the same index) and *computational hiding* (i.e., an honestly-generated commitment σ and set of openings $\{r_i, \pi_i\}_{i\in I}$ should hide all unopened bits r_j for $j \notin I$ from any computationally-bounded adversary). Quach et al. show that an HBG with these properties can be combined directly with a NIZK in the hidden-bits model to obtain a computational NIZK proof in the CRS model. If the HBG is in the (malicious) designated-verifier model, then so is the resulting NIZK.

In this work, we extend this framework by introducing the notion of a dual-mode HBG where the CRS can be generated in one of two modes: a *binding* mode where the HBG satisfies statistical binding (as in [QRW19]) and a *hiding* mode where the HBG satisfies a stronger notion of *statistical hiding* (i.e., the unopened bits are statistically hidden given the CRS, the commitment σ and any subset of opened bits $\{(r_i, \pi_i)\}_{i\in I}$). In our case, we impose an even stronger equivocation property in the hiding mode: namely, given any any set of indices $I \subseteq [\rho]$ and any assignment $r_I \in \{0,1\}^{|I|}$ to that set, it is possible to simulate a commitment σ and a set of openings $\{\pi_i\}_{i\in I}$ that is statistically indistinguishable from the output of the honest generator. This allows us to directly argue *adaptive* and *multi-theorem*[6] statistical zero-knowledge for the resulting NIZK construction. We give our formal definition in Section 3, and describe our construction of dual-mode (designated-verifier) NIZKs from dual-mode (designated-verifier) HBGs in

[6] We can also use the transformation from [FLS99] to generically go from single-theorem zero-knowledge to multi-theorem zero-knowledge, but at the expense of making *non-black-box* use of a PRG. Our approach yields a direct construction of multi-theorem zero-knowledge without needing to make non-black-box use of cryptography. We discuss this in greater detail in Remark 2.5.

Section 3.1. In Section 4 and the full version, we show how to construct dual-mode HBGs from the k-Lin, QR, and DCR assumptions.

2 Preliminaries

Throughout this work, we write λ (oftentimes implicitly) to denote the security parameter. For a positive integer $n \in \mathbb{N}$, we write $[n]$ to denote the set $\{1, \ldots, n\}$. We will typically use bold lowercase letters (e.g., $\mathbf{v}, \mathbf{w}$) to denote vectors and bold uppercase letters (e.g., $\mathbf{A}, \mathbf{B}$) to denote matrices. For a vector $\mathbf{v} \in \mathbb{Z}_p^n$, we will use non-boldface letters to refer to its components; namely, we write $\mathbf{v} = (v_1, \ldots, v_n)$. For a (sorted) set of indices $I = \{i_1, \ldots, i_m\} \subseteq [n]$, we write $\mathbf{v}_I$ to denote the sub-vector $(v_{i_1}, \ldots, v_{i_m})$.

We say that a function f is negligible in λ, denoted $\mathsf{negl}(\lambda)$, if $f(\lambda) = o(1/\lambda^c)$ for all $c \in \mathbb{N}$. We write $\mathsf{poly}(\lambda)$ to denote a function bounded by a fixed polynomial in λ. We say an event happens with negligible probability if the probability of the event happening is negligible, and that it happens with overwhelming probability if its complement occurs with negligible probability. We say that an algorithm is efficient if it runs in probabilistic polynomial-time in the length of its inputs. We say that two families of distributions $\mathcal{D}_1 = \{\mathcal{D}_{1,\lambda}\}_{\lambda \in \mathbb{N}}$ and $\mathcal{D}_2 = \{\mathcal{D}_{2,\lambda}\}_{\lambda \in \mathbb{N}}$ are computationally indistinguishable if no efficient adversary can distinguish samples from $\mathcal{D}_1$ and $\mathcal{D}_2$ except with negligible probability, and we denote this by writing $\mathcal{D}_1 \overset{c}{\approx} \mathcal{D}_2$. For two distributions $\mathcal{D}_1$, $\mathcal{D}_2$, we write $\Delta(\mathcal{D}_1, \mathcal{D}_2)$ to denote the statistical distance between $\mathcal{D}_1$ and $\mathcal{D}_2$. We write $\mathcal{D}_1 \overset{s}{\approx} \mathcal{D}_2$ to denote that $\mathcal{D}_1$ and $\mathcal{D}_2$ are statistically indistinguishable: namely, that $\Delta(\mathcal{D}_1, \mathcal{D}_2) = \mathsf{negl}(\lambda)$. For a finite set S, we write $x \overset{\mathrm{R}}{\leftarrow} S$ to denote that x is sampled uniformly at random from S. For a distribution $\mathcal{D}$, we write $x \leftarrow \mathcal{D}$ to denote that x is sampled from $\mathcal{D}$. We review additional preliminaries in the full version.

2.1 NIZKs in the Hidden-Bits Model

In this section, we recall the notion of a NIZK in the hidden-bits model [FLS99]. Our presentation is adapted from the description from [QRW19, CH19, KNYY19a].

Definition 2.1 (NIZKs in the Hidden-Bits Model). *Let $\mathcal{L} \subseteq \{0,1\}^n$ be an* NP *language associated with an* NP *relation $\mathcal{R}$ with $n = n(\lambda)$. A non-interactive zero-knowledge proof in the hidden-bits model for $\mathcal{L}$ consists of a tuple $\Pi_{\mathsf{HBM}} = (\mathsf{Prove}, \mathsf{Verify})$ and a parameter $\rho = \rho(\lambda, n)$ with the following properties:*

- $\mathsf{Prove}(1^\lambda, r, x, w) \to (I, \pi)$*: On input the security parameter λ, a string $r \in \{0,1\}^\rho$, a statement $x \in \{0,1\}^n$ and a witness w, this algorithm outputs a set of indices $I \subseteq [\rho]$ and a proof π.*
- $\mathsf{Verify}(1^\lambda, I, r_I, x, \pi) \to \{0,1\}$*: On input the security parameter λ, a subset $I \subseteq [\rho]$, a string $r_I \in \{0,1\}^{|I|}$, a statement $x \in \{0,1\}^n$ and a proof π, the verification algorithm outputs a bit $b \in \{0,1\}$.*

Moreover, Π_{HBM} satisfies the following properties:

– ***Completeness:*** *For all* $(x, w) \in \mathcal{R}$ *and* $r \in \{0,1\}^\rho$,

$$\Pr[(I, \pi) \leftarrow \mathsf{Prove}(1^\lambda, r, x, w) : \mathsf{Verify}(1^\lambda, I, r_I, x, \pi) = 1] = 1.$$

– ***Statistical soundness:*** *For all unbounded provers* $\mathcal{P}^*$*, we have that for* $r \xleftarrow{R} \{0,1\}^\rho$ *and* $(x, \pi, I) \leftarrow \mathcal{P}^*(1^\lambda, r)$,

$$\Pr[x \notin \mathcal{L} \ \wedge \ \mathsf{Verify}(1^\lambda, I, r_I, x, \pi) = 1] = \mathsf{negl}(\lambda).$$

We will oftentimes refer to the above probability as the soundness error*.*

– ***Perfect zero-knowledge:*** *There exists an efficient simulator* $\mathcal{S}$ *such that for all unbounded verifiers* $\mathcal{V}^*$*, if we take* $(x, w) \leftarrow \mathcal{V}^*(1^\lambda)$*,* $r \xleftarrow{R} \{0,1\}^\rho$*,* $(I, \pi) \leftarrow \mathsf{Prove}(1^\lambda, r, x, w)$*, and* $(\widetilde{I}, \widetilde{r_I}, \widetilde{\pi}) \leftarrow \mathcal{S}(1^\lambda, x)$*, and moreover if* $\mathcal{R}(x, w) = 1$*, then the following two distributions are identically distributed:*

$$(I, r_I, \pi) \equiv (\widetilde{I}, \widetilde{r_I}, \widetilde{\pi}).$$

Theorem 2.2 (NIZKs in the Hidden-Bits Model [FLS99]). *For any* $\varepsilon > 0$*, every language* $\mathcal{L} \in \mathsf{NP}$ *has a NIZK in the hidden-bits model with soundness error* ε *and relying on a hidden-bits string of length* $\rho = \mathsf{poly}(n, \log(1/\varepsilon))$*.*

2.2 Designated-Verifier NIZKs and Dual-Mode NIZKs

We now review the notion of a *reusable* designated-verifier NIZK (DV-NIZK). Namely, we require that the same common reference string and verification state can be *reused* to prove and verify many statements without compromising either soundness or zero-knowledge. As in [LQR+19], we use the fine-grained notion with separate setup and key-generation algorithms. The setup algorithm samples the common reference string (CRS) while the key-generation algorithm generates a public key (used to generate proofs) along with a secret key (used to verify proofs). We allow the same CRS to be *reusable* by many verifiers, who each generate their own public/secret key-pairs. In the traditional notion of DV-NIZKs, the setup and key-generation algorithms would be combined into a single algorithm that outputs the CRS (which would include the public proving key) along with a secret verification key.

Definition 2.3 (Designated-Verifier NIZK). *Let* $\mathcal{L} \subseteq \{0,1\}^n$ *be an* NP *language associated with an* NP *relation* $\mathcal{R}$ *with* $n = n(\lambda)$*. A reusable* designated-verifier non-interactive zero-knowledge (DV-NIZK) proof for $\mathcal{L}$ *consists of a tuple of efficient algorithms* $\Pi_{\mathsf{dvNIZK}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Prove}, \mathsf{Verify})$ *with the following properties:*

– $\mathsf{Setup}(1^\lambda) \to \mathsf{crs}$*: On input the security parameter* λ*, this algorithm outputs a common reference string* crs*. If* Setup *outputs a* uniformly random string*, we say that the scheme is in the* common random string *model.*

– $\mathsf{KeyGen}(\mathsf{crs}) \to (\mathsf{pk}, \mathsf{sk})$*: On input the common reference string* crs*, the key-generation algorithm outputs a public key* pk *and a secret key* sk*.*

– $\mathsf{Prove}(\mathsf{crs}, \mathsf{pk}, x, w) \to \pi$*: On input the common reference string* crs*, a public key* pk*, a statement* $x \in \{0,1\}^n$*, and a witness* w*, this algorithm outputs a proof* π*.*
– $\mathsf{Verify}(\mathsf{crs}, \mathsf{sk}, x, \pi) \to \{0,1\}$*: On input the common reference string* crs*, a secret verification key* sk*, a statement* x*, and a proof* π*, the verification algorithm outputs a bit* $b \in \{0,1\}$*.*

Moreover, Π_{dvNIZK} *should satisfy the following properties:*

– ***Completeness:*** *For all* $(x, w) \in \mathcal{R}$*, and taking* $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$*,* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{crs})$*,*

$$\Pr\left[\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, \mathsf{pk}, x, w) : \mathsf{Verify}(\mathsf{crs}, \mathsf{sk}, x, \pi) = 1\right] = 1.$$

– ***(Statistical) soundness:*** *We consider two variants of soundness:*
  • ***Non-adaptive soundness:*** *For all* $x \notin \mathcal{L}$ *and all polynomials* $q = q(\lambda)$*, and all unbounded adversaries* $\mathcal{A}$ *making at most* q *verification queries, and sampling* $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$*,* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{crs})$*, we have that*

$$\Pr\left[\pi \leftarrow \mathcal{A}^{\mathsf{Verify}(\mathsf{crs},\mathsf{sk},\cdot,\cdot)}(1^\lambda, \mathsf{crs}, \mathsf{pk}, x) : \mathsf{Verify}(\mathsf{crs}, \mathsf{sk}, x, \pi) = 1\right] = \mathsf{negl}(\lambda).$$

  • ***Adaptive soundness:*** *For all polynomials* $q = q(\lambda)$ *and all unbounded adversaries* $\mathcal{A}$ *making at most* q *verification queries, and sampling* $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$*,* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{crs})$*, we have that*

$$\Pr\Big[(x, \pi) \leftarrow \mathcal{A}^{\mathsf{Verify}(\mathsf{crs},\mathsf{sk},\cdot,\cdot)}(1^\lambda, \mathsf{crs}, \mathsf{pk}) : x \notin \mathcal{L} \ \wedge \ \mathsf{Verify}(\mathsf{crs}, \mathsf{sk}, x, \pi) = 1\Big] = \mathsf{negl}(\lambda).$$

  We also define the corresponding notions of computational soundness *where the above properties only need to hold against efficient adversaries* $\mathcal{A}$*.*
– ***(Statistical) zero-knowledge:*** *For all polynomials* $q = q(\lambda)$ *and all unbounded adversaries* $\mathcal{A}$ *making at most* q *oracle queries, there exists an efficient simulator* $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ *such that*

$$\left|\Pr[\mathcal{A}^{\mathcal{O}_0(\mathsf{crs},\mathsf{pk},\cdot,\cdot)}(\mathsf{crs}, \mathsf{pk}, \mathsf{sk}) = 1] - \Pr[\mathcal{A}^{\mathcal{O}_1(\mathsf{st}_\mathcal{S},\cdot,\cdot)}(\widetilde{\mathsf{crs}}, \widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}}) = 1]\right| = \mathsf{negl}(\lambda),$$

  where $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$*,* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{crs})$ *and* $(\mathsf{st}_\mathcal{S}, \widetilde{\mathsf{crs}}, \widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}}) \leftarrow \mathcal{S}_1(1^\lambda)$*, the oracle* $\mathcal{O}_0(\mathsf{crs}, \mathsf{pk}, x, w)$ *outputs* $\mathsf{Prove}(\mathsf{crs}, \mathsf{pk}, x, w)$ *if* $\mathcal{R}(x, w) = 1$ *and* $\perp$ *otherwise, and the oracle* $\mathcal{O}_1(\mathsf{st}_\mathcal{S}, x, w)$ *outputs* $\mathcal{S}_2(\mathsf{st}_\mathcal{S}, x)$ *if* $\mathcal{R}(x, w) = 1$ *and* $\perp$ *otherwise. Similar to soundness, we also consider* computational zero-knowledge *where the above property only needs to hold against efficient adversaries* $\mathcal{A}$*.*

Definition 2.4 (Publicly-Verifiable NIZKs). *A NIZK* Π_{NIZK} *is* publicly-verifiable *if the secret key output by* KeyGen *is empty. In this case, we can combine the* Setup *and* KeyGen *algorithms into a single algorithm that just outputs the*

CRS, and there is no notion of separate public/secret keys pk *and* sk*. Both the* Prove *and* Verify *algorithms just take* crs *as input. We can define all of the properties analogously. In the publicly-verifiable setting, we do not need to provide the prover a separate verification oracle in the soundness game.*

Remark 2.5 (Single-Theorem vs. Multi-Theorem Zero-Knowledge). The zero-knowledge property in Definition 2.3 is *multi-theorem* in the sense that the adversary can see proofs of multiple statements. We can consider a weaker notion of single-theorem zero-knowledge where the adversary can only see a proof on a single (adaptively-chosen) statement. Previously, Feige et al. [FLS99] showed how to generically compile a single-theorem NIZK into a multi-theorem NIZK using a PRG. This transformation also applies in the designated-verifier setting [QRW19, CH19, KNYY19a]. One limitation of the [FLS99] transformation is that it requires making *non-black-box* use of a PRG. The constructions we present in this work directly achieve multi-theorem zero-knowledge without needing to go through the [FLS99] transformation. As such, our constructions do *not* require making non-black-box use of any cryptographic primitives.

Malicious DV-NIZKs. We also consider the notion of a malicious designated-verifier NIZK (MDV-NIZK) from [QRW19] where zero-knowledge holds even when the public key pk is chosen maliciously. In this case, the only trusted setup that we require is generating the common reference string (or, in some cases, a common random string), which can be reused by many verifiers.

We recall the formal definition in the full version.

Dual-mode DV-NIZKs. Next, we recall the formal definition of a dual-mode (DV)-NIZK [GOS06, GOS12].

Definition 2.6 (Dual-Mode Designated-Verifier NIZK). *A dual-mode DV-NIZK* $\Pi_{\mathsf{dvNIZK}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Prove}, \mathsf{Verify})$ *is a DV-NIZK with the following additional properties:*

- ***Dual-mode:*** *The* Setup *algorithm takes an additional argument* $\mathsf{mode} \in \{\mathsf{binding}, \mathsf{hiding}\}$*, and outputs a common reference string* crs*.*
- ***CRS indistinguishability:*** *The common reference string output by the two modes are computationally indistinguishable:*

$$\mathsf{Setup}(1^\lambda, \mathsf{binding}) \overset{c}{\approx} \mathsf{Setup}(1^\lambda, \mathsf{hiding}).$$

- ***Statistical soundness in binding mode:*** *If* $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{binding})$*, the designated-verifier NIZK satisfies statistical soundness.*
- ***Statistical zero-knowledge in hiding mode:*** *If* $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{hiding})$*, the designated-verifier NIZK satisfies statistical zero-knowledge.*

We define a dual mode MDV-NIZK analogously by requiring the stronger property of statistical zero-knowledge against malicious verifiers in hiding mode.

Remark 2.7 (Dual-Mode Designated-Verifier NIZKs). Let $\Pi_{\mathsf{dvNIZK}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Prove}, \mathsf{Verify})$ be a dual-mode DV-NIZK for a language $\mathcal{L} \subseteq \{0,1\}^n$. Then, the following properties hold:

- When the CRS is generated in binding mode, Π_{dvNIZK} satisfies statistical soundness and computational zero-knowledge (i.e., Π_{dvNIZK} is a "computational DV-NIZK proof").
- When the CRS is generated in hiding mode, Π_{dvNIZK} satisfies *non-adaptive* computational soundness and statistical zero-knowledge (i.e., Π_{dvNIZK} is a "statistical DV-NIZK argument").
- If Π_{dvNIZK} is a dual-mode MDV-NIZK, then the zero-knowledge properties in each of the above instantiations also hold against malicious verifiers.

The first two properties follow from CRS indistinguishability and the corresponding statistical properties of Π_{dvNIZK} in the two modes. Note though that even if Π_{dvNIZK} satisfies adaptive soundness in binding mode, we do not know how to argue *adaptive soundness* for Π_{dvNIZK} in hiding mode. At a high-level, this is because in the definition of adaptive soundness, checking whether the adversary succeeded or not requires deciding whether the statement x output by the adversary is contained in the language $\mathcal{L}$ or not. Unless $\mathsf{NP} \subseteq \mathsf{P/poly}$, this is not an efficiently-checkable property in general, and as such, we are not able to directly argue adaptive soundness of the construction. We refer to [AF07] for more discussion on the challenges of using black-box reductions to argue adaptive soundness for statistical NIZK arguments.

Remark 2.8 (Adaptive Soundness via Complexity Leveraging). Using complexity leveraging [BB04] and relying on a sub-exponential hardness assumption (as in [GOS06, GOS12]), we can show that non-adaptive soundness implies adaptive soundness. A direct application of complexity leveraging to a dual-mode NIZK yields an adaptively-sound statistical NIZK argument for proving statements of a priori bounded length $n = n(\lambda)$. Using the method from [QRW19, §7], we can also obtain adaptive soundness for statements with arbitrary polynomial length, but still at the expense of a subexponential hardness assumption.

3 Dual-Mode Hidden-Bits Generators and Dual-Mode DV-NIZKs

In this section, we formally define a dual-mode hidden-bits generator. Our definition extends the notion of a hidden-bits generator from [QRW19] (and the similar notion of a designated-verifier PRG from [CH19]). Our definition differs from that in [QRW19] in the following respects:

- **Dual mode:** We require that the common reference string for the hidden-bits generator can be generated in two computationally indistinguishable modes: a *binding* mode where the commitment statistically binds to a sequence of hidden bits, and a *hiding* mode where the commitment (and the openings to any subset of the bits) statistically hide the remaining bits.

– **Statistical simulation in hiding mode.** Minimally, our hiding property requires that the commitment and openings to any subset of the bits output by the HBG *statistically* hide the unopened bits. Here, we require an even stronger *simulation* property where there is an efficient simulator that can simulate the commitment and openings to any (random) string, given *only* the values of the opened bits. Moreover, we allow the adversary to adaptively choose the subset of bits for which it wants to see openings, and we also allow *multiple* interactions with the simulator. This strong simulation property enables us to directly argue *adaptive* and *multi-theorem statistical zero-knowledge* for our NIZK constructions (Section 3.1).[7]

Definition 3.1 (Dual-Mode Hidden-Bits Generator). *Let λ be a security parameter and ρ be the output length. Let $\ell = \ell(\lambda, \rho)$ be a polynomial. A dual-mode (designated-verifier) hidden-bits generator (HBG) with commitments of length ℓ consists of a tuple of efficient algorithms $\Pi_{\mathsf{HBG}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{GenBits}, \mathsf{Verify})$ with the following properties:*

- $\mathsf{Setup}(1^\lambda, 1^\rho, \mathsf{mode}) \to \mathsf{crs}$*: On input the security parameter λ, a length ρ, and a mode $\mathsf{mode} \in \{\mathsf{binding}, \mathsf{hiding}\}$, the setup algorithm outputs a common reference string* crs.
- $\mathsf{KeyGen}(\mathsf{crs}) \to (\mathsf{pk}, \mathsf{sk})$*: On input a common reference string* crs*, the key-generation algorithm outputs a public key* pk *and a secret key* sk.
- $\mathsf{GenBits}(\mathsf{crs}, \mathsf{pk}) \to (\sigma, r, \{\pi_i\}_{i\in[\rho]})$*: On input a common reference string* crs *and a public key* pk*, the bit-generation algorithm outputs a commitment $\sigma \in \{0,1\}^\ell$, a string $r \in \{0,1\}^\rho$, and a collection of proofs π_i for $i \in [\rho]$.*
- $\mathsf{Verify}(\mathsf{crs}, \mathsf{sk}, \sigma, i, r_i, \pi_i) \to \{0,1\}$*: On input a common reference string* crs*, a secret key* sk*, a commitment $\sigma \in \{0,1\}^\ell$, an index $i \in [\rho]$, a bit $r_i \in \{0,1\}$, and a proof π_i, the verification algorithm outputs a bit $b \in \{0,1\}$.*

In addition, we require that Π_{HBG} satisfy the following properties:

- ***Correctness:*** *For all integers $\lambda \in \mathbb{N}$, and all polynomials $\rho = \rho(\lambda)$, all indices $i \in [\rho]$ and both modes $\mathsf{mode} \in \{\mathsf{binding}, \mathsf{hiding}\}$, and sampling $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^\rho, \mathsf{mode})$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{crs})$, and $(\sigma, r, \{\pi_i\}_{i\in[\rho]}) \leftarrow \mathsf{GenBits}(\mathsf{crs}, \mathsf{pk})$, we have*

$$\Pr[\mathsf{Verify}(\mathsf{crs}, \mathsf{sk}, \sigma, i, r_i, \pi_i) = 1] = 1.$$

- ***Succinctness:*** *The length ℓ of the commitment depends only on the security parameter and not the length of the output: namely, $\ell = \mathsf{poly}(\lambda)$.*[8]

[7] The previous notion from [QRW19] was only sufficient for single-theorem non-adaptive computational zero-knowledge. Extending to adaptive multi-theorem computational zero-knowledge required imposing additional properties on the underlying NIZK in the hidden-bits model as well as making non-black-box use of cryptographic primitives [FLS99].

[8] We remark that this is a *stronger* requirement than the corresponding requirement in [QRW19], which also allows ℓ to scale sublinearly with ρ. We use this definition because it is conceptually simpler and all of our constructions satisfy this stronger property.

– ***CRS indistinguishability:*** *For all polynomials $\rho = \rho(\lambda)$, we have that*

$$\mathsf{Setup}(1^\lambda, 1^\rho, \mathsf{binding}) \overset{c}{\approx} \mathsf{Setup}(1^\lambda, 1^\rho, \mathsf{hiding}).$$

– ***Statistically binding in binding mode:*** *There exists a (possibly inefficient) deterministic algorithm $\mathsf{Open}(\mathsf{crs}, \sigma)$ such that for all polynomials $\rho = \rho(\lambda)$ and $q = q(\lambda)$ and all unbounded adversaries $\mathcal{A}$ making up to q oracle queries, and sampling $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^\rho, \mathsf{binding})$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{crs})$, $(\sigma^*, i^*, r^*, \pi^*) \leftarrow \mathcal{A}^{\mathsf{Verify}(\mathsf{crs},\mathsf{sk},\cdot,\cdot,\cdot,\cdot)}(1^\lambda, 1^\rho, \mathsf{crs}, \mathsf{pk})$, $r \leftarrow \mathsf{Open}(\mathsf{crs}, \sigma^*)$, we have*

$$\Pr[r_{i^*} \neq r^* \wedge \mathsf{Verify}(\mathsf{crs}, \mathsf{sk}, \sigma^*, i^*, r^*, \pi^*) = 1] = \mathsf{negl}(\lambda).$$

– ***Statistical simulation in hiding mode:*** *For all polynomials $\rho = \rho(\lambda)$, $q = q(\lambda)$, and all unbounded adversaries $\mathcal{A}$ making up to q queries, there exists an efficient simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ such that*

$$\big|\Pr[\mathsf{ExptHide}[\mathcal{A}, \mathcal{S}, 0](1^\lambda, 1^\rho) = 1] - \Pr[\mathsf{ExptHide}[\mathcal{A}, \mathcal{S}, 1](1^\lambda, 1^\rho) = 1]\big| = \mathsf{negl}(\lambda), \quad (3.1)$$

where for a bit $b \in \{0,1\}$, the hiding experiment $\mathsf{ExptHide}[\mathcal{A}, \mathcal{S}, b](1^\lambda, 1^\rho)$ is defined as follows:

- ***Setup phase:*** *If $b = 0$, the challenger samples $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^\rho, \mathsf{hiding})$ and $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{crs})$, and gives $(\mathsf{crs}, \mathsf{pk}, \mathsf{sk})$ to $\mathcal{A}$. If $b = 1$, it samples $(\mathsf{st}_{\mathcal{S}}, \widetilde{\mathsf{crs}}, \widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}}) \leftarrow \mathcal{S}_1(1^\lambda, 1^\rho)$ and gives $(\widetilde{\mathsf{crs}}, \widetilde{\mathsf{pk}}, \widetilde{\mathsf{sk}})$ to $\mathcal{A}$.*
- ***Query phase:*** *The adversary $\mathcal{A}$ can now make up to q challenge queries. On each query, the challenger responds as follows:*
  - *If $b = 0$, the challenger computes $(\sigma, r, \{\pi_i\}_{i\in[\rho]}) \leftarrow \mathsf{GenBits}(\mathsf{crs}, \mathsf{pk})$ and gives r to the adversary. If $b = 1$, it responds with $\widetilde{r} \xleftarrow{R} \{0,1\}^\rho$.*
  - *The adversary specifies a subset $I \subseteq [\rho]$.*
  - *If $b = 0$, then the challenger replies with the pair $(\sigma, \{\pi_i\}_{i\in[I]})$ it sampled above. If $b = 1$, it replies to $\mathcal{A}$ with $(\widetilde{\sigma}, \{\widetilde{\pi}_i\}_{i\in I}) \leftarrow \mathcal{S}_2(\mathsf{st}_{\mathcal{S}}, I, \widetilde{r}_I)$.*
- ***Output phase:*** *At the end of the experiment, the adversary outputs a bit $b \in \{0,1\}$, which is the output of the experiment.*

When the difference in Eq. (3.1) is identically zero, we say that Π_{HBG} satisfies perfect simulation in hiding mode.

Definition 3.2 (Publicly-Verifiable Dual-Mode HBG). *A dual-mode HBG Π_{HBG} is publicly-verifiable if the secret key* sk *output by* KeyGen *is empty. In this case, we can combine the* Setup *algorithm and the* KeyGen *algorithm into a single algorithm that just outputs the* crs*, and there is no notion of separate public/secret keys* pk *and* sk*. The* GenBits *and* Verify *algorithms just take* crs *as input. We define all of the other properties analogously. In the publicly-verifiable setting, we do not need to provide the verification oracle to the adversary in the statistical binding security definition.*

Definition 3.3 (Statistical Simulation for Malicious Keys). *Let* $\Pi_{\mathsf{HBG}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{GenBits}, \mathsf{Verify})$ *be a hidden-bits generator. We say that* Π_{HBG} *satisfies* statistical simulation for malicious keys *if it satisfies the following simulation property (where the adversary chooses* pk*) in hiding mode:*

- ***Statistical simulation for malicious keys:*** *For all polynomials* $\rho = \rho(\lambda)$*,* $q = q(\lambda)$*, and all unbounded adversaries* $\mathcal{A}$ *making up to* q *queries, there exists an efficient simulator* $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ *such that*

$$\Big|\Pr[\mathsf{ExptHide}^*[\mathcal{A}, \mathcal{S}, 0](1^\lambda, 1^\rho) = 1] - \Pr[\mathsf{ExptHide}^*[\mathcal{A}, \mathcal{S}, 1](1^\lambda, 1^\rho) = 1]\Big| = \mathsf{negl}(\lambda),$$

  where for a bit $b \in \{0,1\}$*, the hiding experiment* $\mathsf{ExptHide}^*[\mathcal{A}, \mathcal{S}, b](1^\lambda, 1^\rho)$ *is defined to be* $\mathsf{ExptHide}[\mathcal{A}, \mathcal{S}, b](1^\lambda, 1^\rho)$ *with the following differences:*
  - ***Setup phase:*** *If* $b = 0$*, the challenger samples* $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, 1^\rho, \mathsf{hiding})$ *and gives* crs *to* $\mathcal{A}$*. If* $b = 1$*, the challenger samples* $(\mathsf{st}_{\mathcal{S}}, \widetilde{\mathsf{crs}}) \leftarrow \mathcal{S}_1(1^\lambda, 1^\rho)$ *and gives* $\widetilde{\mathsf{crs}}$ *to* $\mathcal{A}$*. The adversary then chooses a public key* pk*.*
  - ***Query phase:*** *Same as in* $\mathsf{ExptHide}[\mathcal{A}, \mathcal{S}, b]$*, except when* $b = 1$*, the challenger also provides the (adversarially-chosen) public key* pk *to the simulator. In other words, when* $b = 1$*, the challenger's reply to* $\mathcal{A}$ *is computed as* $(\widetilde{\sigma}, \{\widetilde{\pi}_i\}_{i \in I}) \leftarrow \mathcal{S}_2(\mathsf{st}_{\mathcal{S}}, \mathsf{pk}, I, \widetilde{r}_I)$*.*
  - ***Output phase:*** *Same as in* $\mathsf{ExptHide}[\mathcal{A}, \mathcal{S}, b]$*.*

3.1 Dual-Mode DV-NIZK from Dual-Mode HBG

In this section, we give our construction of a dual-mode designated-verifier NIZK from a dual-mode designated-verifier HBG and a NIZK in the hidden-bits model. Our generic construction is essentially the same as the corresponding construction from [QRW19]. We do rely on a different argument to show adaptive, multi-theorem statistical zero-knowledge, and in particular, we appeal to the statistical simulation property of our dual-mode HBG that we introduced in Definition 3.1.

Construction 3.4 (Dual-Mode DV-NIZK from Dual-Mode HBG). Let $\mathcal{L} \subseteq \{0,1\}^n$ be an NP language with associated NP relation $\mathcal{R}$. We rely on the following building blocks:

- Let $\Pi_{\mathsf{HBM}} = (\mathsf{HBM.Prove}, \mathsf{HBM.Verify})$ be a NIZK in the hidden-bits model for $\mathcal{L}$, and let $\rho = \rho(\lambda)$ be the length of the hidden-bits string for Π_{HBM}.
- Let $\Pi_{\mathsf{HBG}} = (\mathsf{HBG.Setup}, \mathsf{HBG.KeyGen}, \mathsf{HBG.GenBits}, \mathsf{HBG.Verify})$ be a hidden-bits generator with commitments of length $\ell = \ell(\lambda, \rho)$, where λ is the security parameter and ρ is the output length of the generator.

We construct a dual-mode DV-NIZK $\Pi_{\mathsf{dvNIZK}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Prove}, \mathsf{Verify})$ for $\mathcal{L}$ as follows:

- $\mathsf{Setup}(1^\lambda, \mathsf{mode}) \to \mathsf{crs}$: On input λ and $\mathsf{mode} \in \{\mathsf{binding}, \mathsf{hiding}\}$, sample $s \xleftarrow{R} \{0,1\}^\rho$. Then, run $\mathsf{crs}_{\mathsf{HBG}} \leftarrow \mathsf{HBG.Setup}(1^\lambda, 1^\rho, \mathsf{mode})$, and output $\mathsf{crs} = (\lambda, s, \mathsf{crs}_{\mathsf{HBG}})$.

- $\mathsf{KeyGen}(\mathsf{crs}) \to (\mathsf{pk}, \mathsf{sk})$: On input $\mathsf{crs} = (\lambda, s, \mathsf{crs}_{\mathsf{HBG}})$, the key-generation algorithm runs $(\mathsf{pk}_{\mathsf{HBG}}, \mathsf{sk}_{\mathsf{HBG}}) \leftarrow \mathsf{HBG.KeyGen}(\mathsf{crs}_{\mathsf{HBG}})$ and outputs $\mathsf{pk} = \mathsf{pk}_{\mathsf{HBG}}$ and $\mathsf{sk} = \mathsf{sk}_{\mathsf{HBG}}$.
- $\mathsf{Prove}(\mathsf{crs}, \mathsf{pk}, x, w) \to \pi$: On input $\mathsf{crs} = (\lambda, s, \mathsf{crs}_{\mathsf{HBG}})$, $\mathsf{pk} = \mathsf{pk}_{\mathsf{HBG}}$, $x \in \{0,1\}^n$, and w, compute $(\sigma, r, \{\pi_{\mathsf{HBG},i}\}_{i\in[\rho]}) \leftarrow \mathsf{HBG.GenBits}(\mathsf{crs}_{\mathsf{HBG}}, \mathsf{pk}_{\mathsf{HBG}})$, and an HBM proof $(I, \pi_{\mathsf{HBM}}) \leftarrow \mathsf{HBM.Prove}(1^\lambda, r \oplus s, x, w)$. Output $\pi = (\sigma, I, r_I, \{\pi_{\mathsf{HBG},i}\}_{i\in I}, \pi_{\mathsf{HBM}})$.
- $\mathsf{Verify}(\mathsf{crs}, \mathsf{sk}, x, \pi)$: On input $\mathsf{crs} = (\lambda, s, \mathsf{crs}_{\mathsf{HBG}})$, $\mathsf{sk} = \mathsf{sk}_{\mathsf{HBG}}$, $x \in \{0,1\}^n$, and the proof $\pi = (\sigma, I, r_I, \{\pi_{\mathsf{HBG},i}\}_{i\in I}, \pi_{\mathsf{HBM}})$, output 1 if $\mathsf{HBM.Verify}(1^\lambda, I, r_I \oplus s_I, x, \pi_{\mathsf{HBM}}) = 1$ and $\mathsf{HBG.Verify}(\mathsf{crs}_{\mathsf{HBG}}, \mathsf{sk}_{\mathsf{HBG}}, \sigma, i, r_i, \pi_{\mathsf{HBG},i}) = 1$ for all $i \in I$. Otherwise, output 0.

Theorem 3.5 (Completeness). *If Π_{HBM} is complete and Π_{HBG} is correct, then Π_{dvNIZK} from Construction 3.4 is complete.*

Proof. Take any $\mathsf{mode} \in \{\mathsf{binding}, \mathsf{hiding}\}$, and sample $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda, \mathsf{mode})$, $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{crs})$. Here, $\mathsf{crs} = (\lambda, s, \mathsf{crs}_{\mathsf{HBG}})$, $\mathsf{pk} = \mathsf{pk}_{\mathsf{HBG}}$, and $\mathsf{sk} = \mathsf{sk}_{\mathsf{HBG}}$. Take any statement $(x, w) \in \mathcal{R}$, and let $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, \mathsf{pk}, x, w)$. Then $\pi = (\sigma, I, r_I, \{\pi_{\mathsf{HBG},i}\}_{i\in I}, \pi_{\mathsf{HBM}})$. Consider the behavior of $\mathsf{Verify}(\mathsf{crs}, \mathsf{sk}, x, \pi)$. By correctness of Π_{HBG}, $\mathsf{HBG.Verify}(\mathsf{crs}_{\mathsf{HBG}}, \mathsf{sk}_{\mathsf{HBG}}, \sigma, i, r_i, \pi_{\mathsf{HBG},i}) = 1$ for all $i \in I$. By completeness of Π_{HBM}, $\mathsf{HBM.Verify}(1^\lambda, I, r_I \oplus s_I, x, w) = 1$, and the verifier accepts. □

Theorem 3.6 (CRS Indistinguishability). *If Π_{HBG} satisfies CRS indistinguishability, then Π_{dvNIZK} from Construction 3.4 satisfies CRS indistinguishability.*

Proof. The CRS in Construction 3.4 consists of a tuple $(\lambda, s, \mathsf{crs}_{\mathsf{HBG}})$. In both modes, the first two components are identically distributed, and $\mathsf{crs}_{\mathsf{HBG}}$ is computationally indistinguishable by CRS indistinguishability of Π_{HBG}. □

Theorem 3.7 (Statistical Soundness in Binding Mode). *If Π_{HBM} is statistically sound with soundness error $\varepsilon(\lambda)$, Π_{HBG} is statistically binding in binding mode, and $2^\ell \cdot \varepsilon = \mathsf{negl}(\lambda)$ then Π_{dvNIZK} from Construction 3.4 satisfies adaptive statistical soundness.*

The proof of Theorem 3.7 is very similar to the corresponding proof of adaptive statistical soundness from [QRW19]. We include it in the full version.

Theorem 3.8 (Statistical Zero-Knowledge in Hiding Mode). *If Π_{HBM} satisfies statistical (resp., perfect) zero-knowledge and Π_{HBG} provides statistical (resp., perfect) simulation in hiding mode, then Π_{dvNIZK} from Construction 3.4 satisfies statistical (resp., perfect) zero-knowledge in hiding mode.*

We give the proof of Theorem 3.8 in the full version.

Theorem 3.9 (Statistical Zero-Knowledge against Malicious Verifiers). *If Π_{HBM} satisfies statistical zero-knowledge and Π_{HBG} provides statistical simulation for malicious keys, then Construction 3.4 is a MDV-NIZK. Namely, Construction 3.4 satisfies statistical zero-knowledge against malicious verifiers in hiding mode.*

The proof of Theorem 3.9 follows from a similar argument as Theorem 3.8 and is included in the full version.

4 Dual-Mode HBGs from the k-Lin Assumption

In this section, we show how to construct dual-mode hidden-bits generators from the k-Lin assumption. We begin with a basic construction from the k-Lin assumption (Section 4.1) and then show how to extend it to achieve public verifiability in a pairing group (Section 4.2) as well as how to achieve security against malicious verifiers in a pairing-free group (Section 4.3). In the full version, we also show how to construct dual-mode HBGs from the QR and DCR assumptions.

4.1 Dual-Mode Hidden-Bits Generator from k-Lin

In this section, we show how to construct a dual-mode hidden-bits generator from the k-linear (k-Lin) assumption [BBS04, HK07, Sha07, EHK+13] over *pairing-free* groups for any $k \geq 1$. We note that the 1-Lin assumption is precisely the decisional Diffie-Hellman (DDH) assumption. We begin by recalling some basic notation.

Notation. Throughout this section, we will work with cyclic groups $\mathbb{G}$ of prime order p. We will use multiplicative notation to denote the group operation. For $x \in \mathbb{Z}_p$, we often refer to g^x as an "encoding" of x. For a matrix $\mathbf{A} \in \mathbb{Z}_p^{n \times m}$, we write $g^{\mathbf{A}} \in \mathbb{G}^{n \times m}$ to denote the matrix of group elements formed by taking the element-wise encoding of each component of $\mathbf{A}$.

Definition 4.1 (Prime-Order Group Generator). *A prime-order group generator algorithm* GroupGen *is an efficient algorithm that on input the security parameter 1^λ outputs a description $\mathcal{G} = (\mathbb{G}, p, g)$ of a prime-order group $\mathbb{G}$ with order p and generator g. Throughout this work, we will assume that $1/p = \mathsf{negl}(\lambda)$.*

Construction 4.2 (Dual-Mode Hidden-Bits Generator from k-Lin). Let GroupGen be a prime-order group generator algorithm. We construct a dual-mode hidden-bits generator (HBG) as follows:

- $\mathsf{Setup}(1^\lambda, 1^\rho, \mathsf{mode}) \to \mathsf{crs}$: First, the setup algorithm samples $\mathcal{G} = (\mathbb{G}, p, g) \leftarrow \mathsf{GroupGen}(1^\lambda)$ and a hash function $H \xleftarrow{R} \mathcal{H}$, where $\mathcal{H}$ is a family of hash functions with domain $\mathbb{G}$ and range $\{0,1\}$. Next, it samples $\mathbf{V} \xleftarrow{R} \mathbb{Z}_p^{(\rho+k)\times k}$ and vectors $\mathbf{w}_1, \ldots, \mathbf{w}_\rho \in \mathbb{Z}_p^{\rho+k}$ as follows:
  - If $\mathsf{mode} = \mathsf{hiding}$, sample $\mathbf{w}_i \xleftarrow{R} \mathbb{Z}_p^{\rho+k}$ for all $i \in [\rho]$.
  - If $\mathsf{mode} = \mathsf{binding}$, sample $\mathbf{s}_i \xleftarrow{R} \mathbb{Z}_p^k$ and set $\mathbf{w}_i \leftarrow \mathbf{V}\mathbf{s}_i$ for all $i \in [\rho]$.

  Output $\mathsf{crs} = (\mathcal{G}, H, g^{\mathbf{V}}, g^{\mathbf{w}_1}, \ldots, g^{\mathbf{w}_\rho})$.
- $\mathsf{KeyGen}(\mathsf{crs}) \to (\mathsf{pk}, \mathsf{sk})$: On input $\mathsf{crs} = (\mathcal{G}, H, g^{\mathbf{V}}, g^{\mathbf{w}_1}, \ldots, g^{\mathbf{w}_\rho})$, the key-generation algorithm samples $a \xleftarrow{R} \mathbb{Z}_p$ and $\mathbf{b}_1, \ldots, \mathbf{b}_\rho \xleftarrow{R} \mathbb{Z}_p^k$. For each $i \in [\rho]$, it sets $\mathbf{z}_i \leftarrow \mathbf{w}_i a + \mathbf{V}\mathbf{b}_i \in \mathbb{Z}_p^{\rho+k}$. It outputs

$$\mathsf{pk} = (g^{\mathbf{z}_1}, \ldots, g^{\mathbf{z}_\rho}) \quad \text{and} \quad \mathsf{sk} = (a, \mathbf{b}_1, \ldots, \mathbf{b}_\rho).$$

– $\mathsf{GenBits}(\mathsf{crs}, \mathsf{pk}) \to (\sigma, r, \{\pi_i\}_{i\in[\rho]})$: On input $\mathsf{crs} = (\mathcal{G}, H, g^{\mathbf{V}}, g^{\mathbf{w}_1}, \ldots, g^{\mathbf{w}_\rho})$ and $\mathsf{pk} = (g^{\mathbf{z}_1}, \ldots, g^{\mathbf{z}_\rho})$, sample $\mathbf{y} \xleftarrow{R} \mathbb{Z}_p^{\rho+k}$ and compute for each $i \in [\rho]$,

$$g^{t_i} \leftarrow g^{\mathbf{y}^\top \mathbf{w}_i} \quad \text{and} \quad g^{u_i} \leftarrow g^{\mathbf{y}^\top \mathbf{z}_i}.$$

Next, let $\sigma = g^{\mathbf{y}^\top \mathbf{V}}$. For each $i \in [\rho]$, set $r_i \leftarrow H(g^{t_i})$ and $\pi_i \leftarrow (g^{t_i}, g^{u_i})$, and output σ, r, and $\{\pi_i\}_{i\in[\rho]}$.

– $\mathsf{Verify}(\mathsf{crs}, \mathsf{sk}, \sigma, i, r_i, \pi_i)$: On input $\mathsf{crs} = (\mathcal{G}, H, g^{\mathbf{V}}, g^{\mathbf{w}_1}, \ldots, g^{\mathbf{w}_\rho})$, the secret key $\mathsf{sk} = (a, \mathbf{b}_1, \ldots, \mathbf{b}_\rho)$, $\sigma = g^{\mathbf{c}^\top}$, $i \in [\rho]$, $r_i \in \{0,1\}$, and $\pi_i = (g^{t_i}, g^{u_i})$, output 1 if $g^{u_i} = (g^{t_i a})(g^{\mathbf{c}^\top \mathbf{b}_i})$ and $r_i = H(g^{t_i})$. Otherwise, output 0.

Correctness and security analysis. We now state the correctness and security theorems for Construction 4.2 and give the proofs in the full version.

Theorem 4.3 (Correctness). *Construction 4.2 is correct.*

Theorem 4.4 (Succinctness). *Construction 4.2 is succinct.*

Theorem 4.5 (CRS Indistinguishability). *Suppose the k-*Lin *assumption holds for* GroupGen*. Then, Construction 3.4 satisfies CRS indistinguishability.*

Theorem 4.6 (Statistical Binding in Binding Mode). *Construction 4.2 satisfies statistical binding in binding mode.*

Theorem 4.7 (Statistical Simulation in Hiding Mode). *If $\mathcal{H}$ satisfies statistical uniformity, then Construction 4.2 satisfies statistical simulation in hiding mode.*

Remark 4.8 (Common Random String in Hiding Mode). Construction 4.2 has the property that in hiding mode, the CRS is a collection of *uniformly* random group elements; in other words, the CRS in hiding mode can be sampled as a common *random* string. In conjunction with Construction 3.4, we obtain a statistical NIZK argument in the common *random* string model (and a computational NIZK proof in the common *reference* string model).

4.2 Publicly-Verifiable Hidden-Bit Generators from Pairings

In this section, we describe a variant of our dual-mode hidden-bits generator from Section 4.1 to obtain a *publicly-verifiable* hidden-bits generator from pairings. Our resulting construction does not give a dual-mode hidden-bits generator. Instead, we obtain a standard HBG (where there is a *single* mode) that satisfies statistical simulation and computational binding. Using an analog of Construction 3.4, this suffices to construct a publicly-verifiable statistical NIZK argument. We refer to the full version for the details. Below, we define the computational binding property we use:

Definition 4.9 (Computational Binding). *A* publicly-verifiable *hidden bits generator $\Pi_{\mathsf{HBG}} = (\mathsf{Setup}, \mathsf{GenBits}, \mathsf{Verify})$ is* computationally binding *if the following property holds:*

– ***Computational binding:*** *There exists an efficient extractor $\mathcal{E} = (\mathcal{E}_1, \mathcal{E}_2)$, where $\mathcal{E}_2$ is deterministic, and for all polynomials $\rho = \rho(\lambda)$, the following two properties hold:*
  - ***CRS indistinguishability:*** *The following distributions are computationally indistinguishable:*

$$\{\mathsf{Setup}(1^\lambda, 1^\rho)\} \stackrel{c}{\approx} \{(\mathsf{st}_\mathcal{E}, \widetilde{\mathsf{crs}}) \leftarrow \mathcal{E}_1(1^\lambda, 1^\rho) : \widetilde{\mathsf{crs}}\}.$$

  - ***Binding:*** *For all efficient adversaries $\mathcal{A}$, and sampling $(\mathsf{st}_\mathcal{E}, \widetilde{\mathsf{crs}}) \leftarrow \mathcal{E}_1(1^\lambda, 1^\rho)$ followed by $(\sigma^*, i^*, r^*, \pi^*) \leftarrow \mathcal{A}(1^\lambda, 1^\rho, \widetilde{\mathsf{crs}})$ and $r \leftarrow \mathcal{E}_2(\mathsf{st}_\mathcal{E}, \sigma^*)$, we have that*

$$\Pr[r_{i^*} \neq r^* \wedge \mathsf{Verify}(\widetilde{\mathsf{crs}}, \sigma^*, i^*, r^*, \pi^*) = 1] = \mathsf{negl}(\lambda).$$

Pairing groups. In this section, we work in (asymmetric) pairing groups. We review the notion of a pairing below. We review the kernel k-linear (k-KerLin) assumption from [MRV15, KW15] in the full version.

Definition 4.10 (Prime-Order Pairing-Group Generator). *A prime-order (asymmetric) pairing group generator algorithm* $\mathsf{PairingGroupGen}$ *is an efficient algorithm that on input the security parameter 1^λ outputs a description $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e)$ of two base groups $\mathbb{G}_1$ (generated by g_1), $\mathbb{G}_2$ (generated by g_2), and a target group $\mathbb{G}_T$, all of prime order p, together with an efficiently-computable mapping $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ (called the "pairing"). Finally, the mapping e is bilinear: for all $x, y \in \mathbb{Z}_p$, $e(g_1^x, g_2^y) = e(g_1, g_2)^{xy}$.*

Notation. For a matrix $\mathbf{A}$, we continue to write $g_1^\mathbf{A}$ and $g_2^\mathbf{A}$ to denote matrices of group elements (over $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively). In addition, if we have two matrices $\mathbf{A} \in \mathbb{Z}^{m \times \ell}$ and $\mathbf{B} \in \mathbb{Z}^{\ell \times n}$, we write $e(g_1^\mathbf{A}, g_2^\mathbf{B})$ to denote the operation that outputs $e(g_1, g_2)^{\mathbf{AB}} \in \mathbb{G}_T^{m \times n}$. In particular, the $(i,j)^{\text{th}}$ entry of $e(g_1^\mathbf{A}, g_2^\mathbf{B})$ is computed as

$$\left[e(g_1^\mathbf{A}, g_2^\mathbf{B})\right]_{i,j} = \prod_{k \in [\ell]} e(g_1^{a_{i,k}}, g_2^{b_{k,j}}).$$

Construction 4.11 (Publicly-Verifiable Hidden-Bits Generator from Pairings). Let $\mathsf{PairingGroupGen}$ be a prime-order bilinear group generator algorithm. We construct a publicly-verifiable hidden-bits generator (HBG) as follow:

– $\mathsf{Setup}(1^\lambda, 1^\rho) \to \mathsf{crs}$: The setup algorithm starts by sampling

$$\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e) \leftarrow \mathsf{PairingGroupGen}(1^\lambda)$$

and a hash function $H \stackrel{\text{R}}{\leftarrow} \mathcal{H}$ where $\mathcal{H}$ is a family of hash functions with domain $\mathbb{G}_1$ and range $\{0,1\}$. Next, it samples a matrix $\mathbf{V} \stackrel{\text{R}}{\leftarrow} \mathbb{Z}_p^{(\rho+k)\times k}$, vectors $\mathbf{w}_1, \ldots, \mathbf{w}_k \stackrel{\text{R}}{\leftarrow} \mathbb{Z}_p^{\rho+k}$, and verification components $\mathbf{a} \stackrel{\text{R}}{\leftarrow} \mathbb{Z}_p^{k+1}$, $\mathbf{B}_1, \ldots, \mathbf{B}_\rho \stackrel{\text{R}}{\leftarrow} \mathbb{Z}_p^{k\times(k+1)}$. In addition, it samples $\mathbf{d} \stackrel{\text{R}}{\leftarrow} \mathbb{Z}_p^k$, and constructs the matrix

$$\mathbf{D} = \left(\frac{\mathrm{diag}(\mathbf{d})}{\mathbf{1}^\mathsf{T}}\right) \in \mathbb{Z}_p^{(k+1)\times k}. \tag{4.1}$$

It computes $\hat{\mathbf{a}}^\mathsf{T} \leftarrow \mathbf{a}^\mathsf{T}\mathbf{D} \in \mathbb{Z}_p^k$, and for each $i \in [\rho]$, it computes $\mathbf{Z}_i \leftarrow \mathbf{w}_i\mathbf{a}^\mathsf{T} + \mathbf{V}\mathbf{B}_i \in \mathbb{Z}_p^{(\rho+k)\times(k+1)}$ and $\hat{\mathbf{B}}_i \leftarrow \mathbf{B}_i\mathbf{D} \in \mathbb{Z}_p^{k\times k}$. It outputs

$$\mathsf{crs} = \left(\mathcal{G}, H, g_1^{\mathbf{V}}, g_2^{\hat{\mathbf{a}}^\mathsf{T}}, g_2^{\mathbf{D}}, \left\{g_1^{\mathbf{w}_i}, g_1^{\mathbf{Z}_i}, g_2^{\hat{\mathbf{B}}_i}\right\}_{i\in[\rho]}\right).$$

– $\mathsf{GenBits}(\mathsf{crs}) \to (\sigma, r, \{\pi_i\}_{i\in[k]})$: On input

$$\mathsf{crs} = \left(\mathcal{G}, H, g_1^{\mathbf{V}}, g_2^{\hat{\mathbf{a}}^\mathsf{T}}, g_2^{\mathbf{D}}, \left\{g_1^{\mathbf{w}_i}, g_1^{\mathbf{Z}_i}, g_2^{\hat{\mathbf{B}}_i}\right\}_{i\in[\rho]}\right),$$

sample $\mathbf{y} \xleftarrow{\mathrm{R}} \mathbb{Z}_p^{\rho+k}$, and compute for each $i \in [\rho]$,

$$g_1^{t_i} \leftarrow g_1^{\mathbf{y}^\mathsf{T}\mathbf{w}_i} \quad \text{and} \quad g_1^{\mathbf{u}_i^\mathsf{T}} \leftarrow g_1^{\mathbf{y}^\mathsf{T}\mathbf{Z}_i}.$$

Next, let $\sigma = g_1^{\mathbf{y}^\mathsf{T}\mathbf{V}}$, and for each $i \in [\rho]$, set $r_i \leftarrow H(g_1^{t_i})$ and $\pi_i = (g_1^{t_i}, g_1^{\mathbf{u}_i^\mathsf{T}})$. Output σ, r, and $\{\pi_i\}_{i\in[\rho]}$.

– $\mathsf{Verify}(\mathsf{crs}, \sigma, i, r_i, \pi_i)$: On input $\mathsf{crs} = \left(\mathcal{G}, H, g_1^{\mathbf{V}}, g_2^{\hat{\mathbf{a}}^\mathsf{T}}, g_2^{\mathbf{D}}, \left\{g_1^{\mathbf{w}_i}, g_1^{\mathbf{Z}_i}, g_2^{\hat{\mathbf{B}}_i}\right\}_{i\in[\rho]}\right)$, $\sigma = g_1^{\mathbf{c}^\mathsf{T}}$, $i \in [\rho]$, $r_i \in \{0,1\}$, and $\pi_i = (g_1^{t_i}, g_1^{\mathbf{u}_i^\mathsf{T}})$, output 1 if

$$e(g_1^{t_i}, g_2^{\hat{\mathbf{a}}^\mathsf{T}}) \cdot e(g_1^{\mathbf{c}^\mathsf{T}}, g_2^{\hat{\mathbf{B}}_i}) = e(g_1^{\mathbf{u}_i^\mathsf{T}}, g_2^{\mathbf{D}}) \tag{4.2}$$

and $r_i = H(g_1^{t_i})$. If either check fails, output 0.

Correctness and security analysis. We now state the correctness and security theorems for Construction 4.11 and provide the proofs in the full version.

Theorem 4.12 (Correctness). *Construction 4.11 is correct.*

Theorem 4.13 (Succinctness). *Construction 4.11 is succinct.*

Theorem 4.14 (Computational Binding). *Suppose* PairingGroupGen *outputs groups* $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ *such that the* k-Lin *assumption holds in* $\mathbb{G}_1$ *and the* k-KerLin *assumption holds in* $\mathbb{G}_2$*. Then, Construction 4.11 satisfies computational binding in binding mode.*

Theorem 4.15 (Statistical Simulation). *If* $\mathcal{H}$ *satisfies statistical uniformity, then Construction 4.11 satisfies statistical simulation.*

4.3 Dual-Mode HBG with Malicious Security from k-Lin

We now show how to modify the k-Lin construction from Section 4.1 (Construction 4.2) to obtain a hidden-bits generator with security against malicious verifiers. Combined with Construction 3.4, this yields a dual-mode MDV-NIZK (Theorem 3.9). We refer to Section 1.2 for a high-level description of our approach.

Construction 4.16 (Dual-Mode HBG with Malicious Security from k-Lin). Let ρ be the output length of the hidden-bits generator. We require the following primitives:

– Let $\mathsf{GroupGen}$ be a prime-order group generator algorithm.
– Let $\ell = 3\rho\lambda$ and define $\mathcal{T}_{\lambda,\ell} := \{S \subseteq [\ell] : |S| = \lambda\}$ to be the set of all subsets of $[\ell]$ that contains exactly λ elements. Let $G\colon \{0,1\}^\kappa \to \mathcal{T}_{\lambda,\ell}^\rho \times \mathbb{Z}_p^{\rho\ell}$ be a PRG with seed length $\kappa = \kappa(\lambda)$. Here, p is the order of the group $\mathbb{G}$ output by $\mathsf{GroupGen}$ (on input 1^λ).

  Constructing the PRG G. It is straightforward to construct a PRG with outputs in $\mathcal{T}_{\lambda,\ell}^\rho \times \mathbb{Z}_p^{\rho\ell}$ from a PRG with outputs in $\{0,1\}^{\rho\lambda\ell(1+\lceil\log p\rceil)}$. To see this, it suffices to give an efficient algorithm that maps from the uniform distribution on $\{0,1\}^{\lambda\ell(1+\lceil\log p\rceil)}$ to a distribution that is statistically close to uniform over $\mathcal{T}_{\lambda,\ell} \times \mathbb{Z}_p^\ell$. Take a string $\gamma \in \{0,1\}^{\lambda\ell(1+\lceil\log p\rceil)}$.
  - The first $\lambda\ell$ bits of γ are interpreted as ℓ blocks of λ-bit indices $i_1, \ldots, i_\ell \in \{0,1\}^\lambda$. These indices specify the set $S \subseteq \mathcal{T}_{\lambda,\ell}$ as follows. First, take $S_0 \leftarrow [\ell]$. For each $j \in [\lambda]$, take s_j to be the $(i_j \bmod |S_{j-1}|)^{\text{th}}$ element of S_{j-1} and define $S_j \leftarrow S_{j-1} \setminus \{s_j\}$. Define $S \leftarrow \{s_1, \ldots, s_\ell\} \in \mathcal{T}_{\lambda,\ell}$.
  - The remaining $\lambda\ell\lceil\log p\rceil$ bits of γ are taken to be the binary representation of a vector $\boldsymbol{\alpha} \in \mathbb{Z}^\ell$, where each component is a $\lambda\lceil\log p\rceil$-bit integer.

  The string $\gamma \in \{0,1\}^{\lambda\ell(1+\lceil\log p\rceil)}$ is mapped onto $(S, \boldsymbol{\alpha} \bmod p) \in \mathcal{T}_{\lambda,\ell} \times \mathbb{Z}_p^\ell$. By construction, this procedure maps from the uniform distribution over $\{0,1\}^{\lambda\ell(1+\lceil\log p\rceil}$ to a distribution that is statistically uniform over $\mathcal{T}_{\lambda,\ell} \times \mathbb{Z}_p^\ell$.

We construct the dual-mode designated-verifier hidden-bits generator with malicious security as follows:

– $\mathsf{Setup}(1^\lambda, 1^\rho, \mathsf{mode}) \to \mathsf{crs}$: Let $\ell' = \rho\ell$. Sample $\mathcal{G} = (\mathbb{G}, p, g) \leftarrow \mathsf{GroupGen}(1^\lambda)$ and $H \xleftarrow{\text{R}} \mathcal{H}$, where $\mathcal{H}$ is a family of hash functions with domain $\mathbb{G}$ and range $\{0,1\}$. Next, it samples $\mathbf{V} \xleftarrow{\text{R}} \mathbb{Z}_p^{(\ell'+k)\times k}$ and $\mathbf{w}_1, \ldots, \mathbf{w}_{\ell'} \in \mathbb{Z}_p^{\ell'+k}$ as follows:
  - If $\mathsf{mode} = \mathsf{hiding}$, sample $\mathbf{w}_i \xleftarrow{\text{R}} \mathbb{Z}_p^{\ell'+k}$ for all $i \in [\ell']$.
  - If $\mathsf{mode} = \mathsf{binding}$, sample $\mathbf{s}_i \xleftarrow{\text{R}} \mathbb{Z}_p^k$ and set $\mathbf{w}_i \leftarrow \mathbf{V}\mathbf{s}_i$ for all $i \in [\ell']$.

  Output $\mathsf{crs} = (\mathcal{G}, H, g^{\mathbf{V}}, g^{\mathbf{w}_1}, \ldots, g^{\mathbf{w}_{\ell'}})$.
– $\mathsf{KeyGen}(\mathsf{crs}) \to (\mathsf{pk}, \mathsf{sk})$: On input $\mathsf{crs} = (\mathcal{G}, H, g^{\mathbf{V}}, g^{\mathbf{w}_1}, \ldots, g^{\mathbf{w}_{\ell'}})$, sample $a \xleftarrow{\text{R}} \mathbb{Z}_p$ and $\mathbf{b}_1, \ldots, \mathbf{b}_{\ell'} \xleftarrow{\text{R}} \mathbb{Z}_p^k$. For each $i \in [\ell']$, compute $\mathbf{z}_i \leftarrow \mathbf{w}_i a + \mathbf{V}\mathbf{b}_i \in \mathbb{Z}_p^{\ell'+k}$ and output

$$\mathsf{pk} = (g^{\mathbf{z}_1}, \ldots, g^{\mathbf{z}_{\ell'}}) \quad \text{and} \quad \mathsf{sk} = (a, \mathbf{b}_1, \ldots, \mathbf{b}_{\ell'}).$$

– $\mathsf{GenBits}(\mathsf{crs}, \mathsf{pk}) \to (\sigma, r, \{\pi_i\}_{i\in[\rho]})$: On input $\mathsf{crs} = (\mathcal{G}, H, g^{\mathbf{V}}, g^{\mathbf{w}_1}, \ldots, g^{\mathbf{w}_{\ell'}})$ and $\mathsf{pk} = (g^{\mathbf{z}_1}, \ldots, g^{\mathbf{z}_{\ell'}})$, sample $\mathbf{y} \xleftarrow{\text{R}} \mathbb{Z}_p^{\ell'+k}$ and compute for each $i \in [\ell']$

$$g^{t_i} \leftarrow g^{\mathbf{y}^\top \mathbf{w}_i} \quad \text{and} \quad g^{u_i} \leftarrow g^{\mathbf{y}^\top \mathbf{z}_i}.$$

  Next, sample a PRG seed $\mathsf{s} \xleftarrow{\text{R}} \{0,1\}^\kappa$ and compute $(\hat{S}_1, \ldots, \hat{S}_\rho, \boldsymbol{\alpha}) \leftarrow G(\mathsf{s})$ where $\hat{S}_i \in \mathcal{T}_{\lambda,\ell}$ for all $i \in [\rho]$ and $\boldsymbol{\alpha} \in \mathbb{Z}_p^{\rho\ell}$. Compute the shifted sets $S_i \leftarrow \{j + \ell \cdot (i-1) \mid j \in \hat{S}_i\}$ for each $i \in [\rho]$. Finally, compute

$$r_i \leftarrow H\left(\prod_{j\in S_i} g^{\alpha_j t_j}\right) \text{ and } \quad \pi_i \leftarrow \{(j, g^{t_j}, g^{u_j})\}_{j\in S_i}.$$

Output $\sigma = (\mathsf{s}, g^{\mathbf{y}^\top \mathbf{V}})$, r, and $\{\pi_i\}_{i\in[\rho]}$.

– $\mathsf{Verify}(\mathsf{crs}, \mathsf{sk}, \sigma, i, r_i, \pi_i)$: On input $\mathsf{crs} = (\mathcal{G}, H, g^{\mathbf{V}}, g^{\mathbf{w}_1}, \ldots, g^{\mathbf{w}_{\ell'}})$, the secret key $\mathsf{sk} = (a, \mathbf{b}_1, \ldots, \mathbf{b}_{\ell'})$, $\sigma = (\mathsf{s}, g^{\mathbf{c}^\top})$, $i \in [\rho]$, $r_i \in \{0,1\}$, and $\pi_i = \{(j, g^{t_j}, g^{u_j})\}_{j\in S}$ for an implicitly-defined set $S \subseteq [\rho\ell]$, the verification algorithm performs the following checks:
  - Compute $(\hat{S}_1, \ldots, \hat{S}_\rho, \boldsymbol{\alpha}) \leftarrow G(\mathsf{s})$ and the shifted set $S_i \leftarrow \{j + \ell \cdot (i-1) \mid j \in \hat{S}_i\}$. It checks that $S = S_i$ and outputs 0 if not.
  - It checks that $g^{u_j} = (g^{t_j a})(g^{\mathbf{c}^\top \mathbf{b}_j})$ for all $j \in S$, and outputs 0 if not.
  - It checks that $r_i = H\big(\prod_{j\in S} g^{\alpha_j t_j}\big)$ and outputs 0 if not.

  If all checks pass, the verification algorithm outputs 1.

Correctness and security analysis. We now state the correctness and security theorems for Construction 4.16 and provide the proofs in the full version.

Theorem 4.17 (Correctness). *Construction 4.16 is correct.*

Theorem 4.18 (Succinctness). *Construction 4.16 is succinct.*

Theorem 4.19 (CRS Indistinguishability). *Suppose the k-Lin assumption holds for GroupGen. Then, Construction 4.16 satisfies CRS indistinguishability.*

Theorem 4.20 (Statistical Binding in Binding Mode). *Construction 4.16 satisfies statistical binding in binding mode.*

Theorem 4.21 (Statistical Simulation in Hiding Mode). *If G is a secure PRG and $\mathcal{H}$ satisfies statistical uniformity, then Construction 4.16 satisfies statistical simulation in hiding mode against malicious verifiers.*

5 Instantiations and Extensions

In this section, we provide the main implications of our framework for constructing statistical (and more generally, dual-mode) NIZKs. In the full version, we describe two simple extensions to augment our NIZKs with additional properties.

Dual-mode MDV-NIZKs. By instantiating Construction 3.4 with a dual-mode malicious designated-verifier hidden-bits generator, we obtain a dual-mode MDV-NIZK (Theorems 3.5, 3.7 and 3.9).

Corollary 5.1 (Dual-Mode MDV-NIZK from k-Lin). *Under the k-Lin assumption over pairing-free groups (for any $k \geq 1$), there exists a statistical MDV-NIZK argument (with non-adaptive soundness) in the common* random *string model, and a computational MDV-NIZK proof (with adaptive soundness) for* NP *in the common* reference *string model.*

Corollary 5.2 (Dual-Mode MDV-NIZK from QR or DCR). *Under the* QR *or* DCR *assumptions, there exists a statistical MDV-NIZK argument (with non-adaptive soundness) and a computational MDV-NIZK proof (with adaptive soundness) for* NP *in the common reference string model.*

Publicly-verifiable statistical NIZK arguments. In the full version, we show how to obtain a publicly-verifiable statistical NIZK argument in the common reference string model using Construction 4.11:

Corollary 5.3 (Publicly-Verifiable Statistical NIZK Argument from Pairings). *Suppose that the k-*Lin *assumption holds in $\mathbb{G}_1$ and the k-*KerLin *assumption holds in $\mathbb{G}_2$ (for any $k \geq 1$) over a pairing group. Then, there exists a publicly-verifiable statistical NIZK argument for* NP *(with non-adaptive soundness) in the common reference string model.*

Acknowledgments

We thanks the anonymous Eurocrypt reviewers for helpful feedback on this work.

References

AF07. Masayuki Abe and Serge Fehr. Perfect NIZK with adaptive soundness. In *TCC*, 2007.

BB04. Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity-based encryption without random oracles. In *EUROCRYPT*, 2004.

BBS04. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *CRYPTO*, 2004.

BCG+19. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In *CRYPTO*, 2019.

BCGI18. Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector OLE. In *ACM CCS*, 2018.

BFM88. Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *STOC*, 1988.

BG10. Zvika Brakerski and Shafi Goldwasser. Circular and leakage resilient public-key encryption under subgroup indistinguishability - (or: Quadratic residuosity strikes back). In *CRYPTO*, 2010.

Blu86. Manuel Blum. How to prove a theorem so no one else can claim it. In *Proceedings of the International Congress of Mathematicians*, volume 1, 1986.

BY92. Mihir Bellare and Moti Yung. Certifying cryptographic tools: The case of trapdoor permutations. In *CRYPTO*, 1992.

CCH+19. Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. Fiat-Shamir: from practice to theory. In *STOC*, 2019.

CDI+19. Melissa Chase, Yevgeniy Dodis, Yuval Ishai, Daniel Kraschewski, Tianren Liu, Rafail Ostrovsky, and Vinod Vaikuntanathan. Reusable non-interactive secure computation. In *CRYPTO*, 2019.

CH19. Geoffroy Couteau and Dennis Hofheinz. Designated-verifier pseudorandom generators, and their applications. In *EUROCRYPT*, 2019.

CHK03. Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In *EUROCRYPT*, 2003.

CKS08. David Cash, Eike Kiltz, and Victor Shoup. The twin Diffie-Hellman problem and applications. In *EUROCRYPT*, 2008.

CL18. Ran Canetti and Amit Lichtenberg. Certifying trapdoor permutations, revisited. In *TCC*, 2018.

CS98. Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *CRYPTO*, 1998.

CS02. Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *EUROCRYPT*, 2002.

DDO+01. Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In *CRYPTO*, 2001.

DFN06. Ivan Damgård, Nelly Fazio, and Antonio Nicolosi. Non-interactive zero-knowledge from homomorphic encryption. In *TCC*, 2006.

DGI+19. Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. Trapdoor hash functions and their applications. In *CRYPTO*, 2019.

DMP87. Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Non-interactive zero-knowledge proof systems. In *CRYPTO*, 1987.

EHK+13. Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge L. Villar. An algebraic framework for Diffie-Hellman assumptions. In *CRYPTO*, 2013.

FLS90. Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *FOCS*, 1990.

FLS99. Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs under general assumptions. *SIAM J. Comput.*, 29(1), 1999.

FS86. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, 1986.

GM82. Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *STOC*, 1982.

GMR89. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1), 1989.

Gol11. Oded Goldreich. Basing non-interactive zero-knowledge on (enhanced) trapdoor permutations: The state of the art. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*. 2011.

GOS06. Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In *EUROCRYPT*, 2006.

GOS12. Jens Groth, Rafail Ostrovsky, and Amit Sahai. New techniques for noninteractive zero-knowledge. *J. ACM*, 59(3), 2012.

GR13. Oded Goldreich and Ron D. Rothblum. Enhancements of trapdoor permutations. *J. Cryptology*, 26(3), 2013.

Gro10. Jens Groth. Short non-interactive zero-knowledge proofs. In *ASIACRYPT*, 2010.

HILL99. Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4), 1999.

HJO+16. Brett Hemenway, Zahra Jafargholi, Rafail Ostrovsky, Alessandra Scafuro, and Daniel Wichs. Adaptively secure garbled circuits from one-way functions. In *CRYPTO*, 2016.

HJR16. Dennis Hofheinz, Tibor Jager, and Andy Rupp. Public-key encryption with simulation-based selective-opening security and compact ciphertexts. In *TCC*, 2016.

HK07. Dennis Hofheinz and Eike Kiltz. Secure hybrid encryption from weakened key encapsulation. In *CRYPTO*, 2007.

HU19. Dennis Hofheinz and Bogdan Ursu. Dual-mode NIZKs from obfuscation. In *ASIACRYPT*, 2019.

KNYY19a. Shuichi Katsumata, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Designated verifier/prover and preprocessing NIZKs from Diffie-Hellman assumptions. In *EUROCRYPT*, 2019.

KNYY19b. Shuichi Katsumata, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Exploring constructions of compact NIZKs from various assumptions. In *CRYPTO*, 2019.

KW15. Eike Kiltz and Hoeteck Wee. Quasi-adaptive NIZK for linear subspaces revisited. In *EUROCRYPT*, 2015.

KW18. Sam Kim and David J. Wu. Multi-theorem preprocessing NIZKs from lattices. In *CRYPTO*, 2018.

LQR$^+$19. Alex Lombardi, Willy Quach, Ron D. Rothblum, Daniel Wichs, and David J. Wu. New constructions of reusable designated-verifier NIZKs. In *CRYPTO*, 2019.

MRV15. Paz Morillo, Carla Ràfols, and Jorge L. Villar. Matrix computational assumptions in multilinear groups. *IACR Cryptology ePrint Archive*, 2015.

Pai99. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, 1999.

PS19. Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for NP from (plain) learning with errors. In *CRYPTO*, 2019.

PsV06. Rafael Pass, Abhi shelat, and Vinod Vaikuntanathan. Construction of a non-malleable encryption scheme from any semantically secure one. In *CRYPTO*, 2006.

QRW19. Willy Quach, Ron D. Rothblum, and Daniel Wichs. Reusable designated-verifier NIZKs for all NP from CDH. In *EUROCRYPT*, 2019.

Sha07. Hovav Shacham. A Cramer-Shoup encryption scheme from the linear assumption and from progressively weaker linear variants. *IACR Cryptology ePrint Archive*, 2007.

SW14. Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *STOC*, 2014.