NIZK from SNARG

Fuyuki Kitagawa[1], Takahiro Matsuda[2], Takashi Yamakawa[1]

[1]NTT Secure Platform Laboratories, Tokyo, Japan
{fuyuki.kitagawa.yh, takashi.yamakawa.ga}@hco.ntt.co.jp
[2]National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, Japan
t-matsuda@aist.go.jp

May 26, 2020

Abstract

We give a construction of a non-interactive zero-knowledge (NIZK) argument for all NP languages based on a succinct non-interactive argument (SNARG) for all NP languages and a one-way function. The succinctness requirement for the SNARG is rather mild: We only require that the proof size be $|\pi| = \mathsf{poly}(\lambda)(|x|+|w|)^c$ for some constant $c < 1/2$, where $|x|$ is the statement length, $|w|$ is the witness length, and λ is the security parameter. Especially, we do not require anything about the efficiency of the verification.

Based on this result, we also give a generic conversion from a SNARG to a zero-knowledge SNARG assuming the existence of CPA secure public-key encryption. For this conversion, we require a SNARG to have efficient verification, i.e., the computational complexity of the verification algorithm is $\mathsf{poly}(\lambda)(|x| + |w|)^{o(1)}$. Before this work, such a conversion was only known if we additionally assume the existence of a NIZK.

Along the way of obtaining our result, we give a generic compiler to upgrade a NIZK for all NP languages with non-adaptive zero-knowledge to one with adaptive zero-knowledge. Though this can be shown by carefully combining known results, to the best of our knowledge, no explicit proof of this generic conversion has been presented.

1 Introduction

A *non-interactive zero-knowledge (NIZK)* argument [BFM88] is a non-interactive argument system that enables a prover to convince a verifier of the truth of an NP statement without revealing any information about its witness. Since it is known that a NIZK in the plain model where no setup is needed exists only for trivial languages [GO94], NIZKs are typically constructed in the common reference string (CRS) model where a trusted party generates a CRS and provides it to both the prover and verifier. In the following, we refer to NIZKs in the CRS model simply as NIZKs. Thus far, NIZKs for all NP languages have been constructed based on various standard assumptions including factoring [FLS99], pairings [CHK07, GOS12], and lattices [PS19]. Besides the theoretical importance on its own, NIZKs have found numerous applications in cryptography including chosen-ciphertext security [NY90, DDN00], leakage- and tamper-resilient cryptography [KV09, DHLW10, DFMV13], advanced types of digital signatures [Cv91, RST01, BMW03], multi-party computation [GMW87], to name a few.

A *succinct non-interactive argument (SNARG)* is another notion of a non-interactive argument, which satisfies *succinctness*, i.e., the proof size is (asymptotically) smaller than the statement size and the witness size. Micali [Mic00] gave a construction of SNARGs for all NP

languages in the random oracle model. On the other hand, Gentry and Wichs [GW11] ruled out a black-box reduction proving the adaptive soundness of a SNARG from any falsifiable assumption in the standard model. Since then, there have been proposed constructions of SNARGs for all NP languages based on non-falsifiable assumptions on pairings [Gro10, GGPR13, Gro16], lattices [BISW17, BISW18][1], or hash functions [BCC+17]. On the application side, SNARGs have natural applications in the context of verifiable computation. They also have been gaining a renewed attention in the context of blockchains (e.g., [BCG+14, BBBF18]).[2]

As mentioned above, there are constructions of NIZKs based on various standard assumptions while there is no known construction of SNARGs based on a standard assumption and there is even a strong impossibility for that. Given this situation, we may think that a SNARG is a stronger primitive than a NIZK. However, it is not known if a SNARG implies a NIZK, and they have been treated as incomparable primitives. For example, Bitansky et al. [BCC+17] gave a generic conversion from a SNARG to a zero-knowledge SNARG by additionally assuming the existence of NIZKs. If a SNARG implies a NIZK, we could drop the additional assumption of the NIZK. Besides, since both NIZKs and SNARGs are important and fundamental primitives that have been well-studied, we believe that it is interesting on its own if we find a new relationship between them.

1.1 Our Results

We give a construction of a NIZK for all NP languages based on a SNARG for all NP languages and a one-way function (OWF). The succinctness requirement for the SNARG is rather mild: We only require that its proof size be $|\pi| = \mathsf{poly}(\lambda)(|x| + |w|)^c$ for some constant $c < 1/2$, where $|x|$ is the statement length, $|w|$ is the witness length, and λ is the security parameter. Especially, we do not require anything about the efficiency of the verification.

Based on this result, we also give a generic conversion from a SNARG to a zero-knowledge SNARG assuming the existence of CPA secure public-key encryption. For this conversion, we require a SNARG to have efficient verification, i.e., the computational complexity of the verification algorithm is $\mathsf{poly}(\lambda)(|x| + |w|)^{o(1)}$ (and thus the proof size is also $|\pi| = \mathsf{poly}(\lambda)(|x| + |w|)^{o(1)}$). Before this work, such a conversion was only known if we additionally assume the existence of a NIZK [BCC+17].

Along the way of obtaining our result, we give a generic compiler to upgrade a NIZK for all NP languages with non-adaptive zero-knowledge to one with adaptive zero-knowledge. Though this can be shown by carefully combining known results, to the best of our knowledge, no explicit proof of this generic conversion has been presented.[3] [4]

We note that we use the adaptive computational soundness as a default notion of soundness for non-interactive arguments in this paper, and our results are proven in this setting. We leave it as an interesting open problem to study if similar implications hold for NIZKs and SNARGs with non-adaptive computational soundness.

To the best of our knowledge, all known constructions of a SNARG in the CRS model satisfies zero-knowledge property from the beginning. Therefore, we do not obtain a concrete

[1] The lattice based constructions are in the designated verifier model where a designated party that holds a verification key can verify proofs.

[2] Actually, what is often used in blockchains is a SNARK [BCC+17], which is a stronger variant of a SNARG that satisfies extractability. We often refer to a SNARK as a SNARG since we do not discuss extractability in this paper.

[3] Dwork and Naor [DN07] showed a similar compiler for a NIZK *proof* in the common *random* string. But their compiler does not work for a NIZK *argument* in the common *reference* string model.

[4] A recent work by Couteau, Katsumata, and Ursu [CKU20] implicitly relies on a similar observation. However, they do not state it in a general form, and they only analyze their specific instantiations.

construction of a NIZK from an assumption that was not known to imply NIZKs by using our result. Nonetheless, it is in general important to stydy generic relationships between different primitives from a theoretical point of view, and we believe that our results contribute to deepening our understanding on the two important and fundamental primitives of NIZKs and SNARGs.

1.2 Technical Overview

In this section, we give an overview for the construction of a NIZK from a SNARG. Once this is done, it is straightforward to obtain a generic conversion from a SNARG to a zero-knowledge SNARG by combining it with the result of [BCC+17].

First, we observe that the succinctness of a SNARG implies that a SNARG proof at least "loses" some information about the witness though it may leak some partial information. Based on this observation, our basic idea is to combine a SNARG with a leakage-resilient primitive [AGV09] whose security holds even if a certain fraction of a secret key is leaked. If the SNARG proof size is small enough, then we may be able to use the security of the leakage-resilient primitive to fully hide the witness considering a SNARG proof as a leakage. For example, suppose that we have a leakage-resilient secret-key encryption (LR-SKE) scheme whose semantic security holds as long as the amount of leakage from the secret key is at most a half of the secret key size. Then, a naive (failed) idea to construct a NIZK is to let a NIZK proof consist of an encryption ct of the witness by the LR-SKE scheme and a SNARG proof proving that there exists a secret key of the LR-SKE scheme that decrypts ct to a valid witness. Soundness of this construction is easy to reduce to the soundness of the SNARG. In addition, if the SNARG is fully succinct, then we can show that the SNARG proof size is at most a half of the secret key size if we set the secret key size of LR-SKE to be sufficiently large. Then, it seems possible to argue that the information of the witness is completely hidden by the security of LR-SKE. However, there is a flaw in the above idea: The security of a LR-SKE scheme holds only if the leakage does not depend on the challenge ciphertext. On the other hand, in the above construction, the SNARG proof clearly depends on the challenge ciphertext ct, and thus we cannot use the security of a LR-SKE scheme. Though the above naive idea fails, this highlights a potential idea of combining a SNARG with a leakage-resilient primitive to obtain a NIZK. Indeed, we implement this idea by modifying the NIZK construction based on the hidden-bits paradigm [FLS99].

NIZK via the Hidden-Bits Paradigm. First, we recall the construction of a NIZK based on the hidden-bits paradigm [FLS99] following the formalization by Quach, Rothblum, and Wichs [QRW19]. Readers familiar with their formalization can safely skip this paragraph. The construction uses two building blocks: a NIZK in the hidden-bit model (HBM-NIZK) and a hidden-bits generator (HBG).

In an HBM-NIZK, a trusted party picks a random string $\rho \in \{0,1\}^k$ and gives it to the prover. Then a prover, who holds a statement x and a witness w, generates a proof π along with a subset $I \subseteq [k]$, which specifies which bits of ρ to be revealed to the verifier. Then, the verifier is given a statement x, a proof π, a subset I, and a string ρ_I that is the substring of ρ on the positions corresponding to I, and accepts or rejects. We require an HBM-NIZK to satisfy two security requirements: soundness and zero-knowledge. Intuitively, soundness requires that no cheating prover can convince the verifier of a false statement x with non-negligible probability, and the zero-knowledge property requires that the verifier learns nothing beyond that x is a true statement. Feige, Lapidot, and Shamir [FLS99] constructed an HBM-NIZK for all NP languages that satisfies these security requirements (without relying on any assumption).

An HBG is a primitive introduced in [QRW19], which consists of the following algorithms:

- $\mathsf{HBG.Setup}(1^\lambda, 1^k)$ generates a CRS crs where k denotes the length of hidden-bits to be generated.
- $\mathsf{HBG.GenBits}(\mathsf{crs})$ generates a succinct commitment com whose length is much shorter than k, "hidden-bits" $r \in \{0,1\}^k$, and a tuple of proofs $\{\pi_i\}_{i\in[k]}$. Intuitively, each π_i can be thought of a certificate of the i-th bit of r.
- $\mathsf{HBG.Verify}(\mathsf{crs}, \mathsf{com}, i, r_i, \pi_i)$ verifies the proof π_i to ensure that the i-th hidden-bit is r_i.

We require an HBG to satisfy two security requirements: binding and hiding. The binding property requires that for any fixed commitment com, there exist "committed bits" $r^* \in \{0,1\}^k$ and no PPT adversary can generate a proof π_i such that $\mathsf{HBG.Verify}(\mathsf{crs}, \mathsf{com}, i, \bar{r_i^*}, \pi_i)$ accepts, where $\bar{r_i^*}$ denotes the negation of r_i^*.[5] Combined with the succinctness of com, this implies that there should be a "sparse" set $\mathcal{V}^{\mathsf{crs}} \in \{0,1\}^k$ (dependent on crs) of size much smaller than 2^k such that no PPT adversary can generate a set of proofs $\{\pi_i\}_{i\in I}$ for bits that are not consistent with any element of $\mathcal{V}^{\mathsf{crs}}$ even if it can control the value of com. The hiding property requires that for any subset $I \subseteq [k]$, no PPT adversary given $\{(r_i, \pi_i)\}_{i\in I}$ can distinguish $r_{\bar{I}}$ from a fresh random string $r'_{\bar{I}} \xleftarrow{\$} \{0,1\}^{|I|}$, where $r_{\bar{I}}$ denotes the substring of r on the positions corresponding to $\bar{I} = [k] \setminus I$.

Combining the above two primitives, Quach et al. [QRW19] constructed a NIZK as follows: The setup algorithm generates a CRS $\mathsf{crs} \xleftarrow{\$} \mathsf{HBG.Setup}(1^\lambda, 1^k)$ of the HBG and a random string $s \xleftarrow{\$} \{0,1\}^k$ and outputs them as a CRS of the NIZK where $k = \mathsf{poly}(\lambda)$ is a parameter that is set appropriately as explained later. Then the prover generates $(\mathsf{com}, r, \{\pi_i\}_{i\in[k]}) \xleftarrow{\$} \mathsf{HBG.GenBits}(\mathsf{crs})$, sets $\rho := r \oplus s$, runs the prover of the underlying HBM-NIZK w.r.t. the hidden-bits ρ to generate (I, π_{hbm}), and outputs $(I, \pi_{\mathsf{hbm}}, \mathsf{com}, r_I, \{\pi_i\}_{i\in I})$ as a proof of the NIZK. Then the verifier runs the verification of the underlying HBG to check the validity of r_I and the verification algorithm of the underlying HBM-NIZK under the revealed hidden-bits $\rho_I := r_I \oplus s_I$.

The security of the above NIZK is argued as follows: For each fixed r, any cheating prover against the above NIZK can be easily converted into a cheating prover against the underlying HBM-NIZK. Moreover, by the binding property of the underlying HBG, the prover has to use r in the subset $\mathcal{V}^{\mathsf{crs}}$ to pass the verification. Then, by taking the union bound, the success probability of a cheating prover against the above NIZK is at most $|\mathcal{V}^{\mathsf{crs}}| \ll 2^k$ times larger than that of a cheating prover against the underlying HBM-NIZK. Thus, by setting k to be sufficiently large so that the success probability of a cheating prover against the underlying HBM-NIZK is at most $|\mathcal{V}^{\mathsf{crs}}|^{-1}\mathsf{negl}(\lambda)$, we can prove the soundness. Intuitively, the zero-knowledge property of the above NIZK is easy to reduce to that of the underlying HBM-NIZK by observing that the hiding property of the underlying HBG ensures that the verifier obtains no information about $r_{\bar{I}}$. We note that this simple reduction works only for *non-adaptive* zero-knowledge where an adversary declares a challenge statement before seeing a CRS. Roughly speaking, this is because in the definition of the hiding property of a HBG, the subset I is fixed before the CRS is chosen whereas an adversary against adaptive zero-knowledge may choose I depending on the CRS. Quach et al. [QRW19] showed that adaptive zero-knowledge can be also proven assuming that the underlying HBM-NIZK satisfies a stronger notion of zero-knowledge called special zero-knowledge. We omit to explain the details since we will show a generic compiler from non-adaptive to adaptive zero-knowledge.

[5] The original definition in [QRW19] required a stronger requirement of *statistical* binding where the property should hold against all computationally unbounded adversaries.

HBG from a SNARG? Our first attempt is to construct an HBG from a SNARG combined with a leakage-resilient weak pseudorandom function (LR-wPRF) [HLWW16]. A (one-bit-output) LR-wPRF is a function family $\mathcal{F} = \{F_K : \{0,1\}^m \to \{0,1\}\}_{K\in\{0,1\}^\kappa}$ such that $(x^*, F_K(x^*))$ for $x^* \xleftarrow{\$} \{0,1\}^m$ looks pseudorandom from an adversary that is given an arbitrary polynomial number of input-output pairs $(x, F_K(x))$ for $x \xleftarrow{\$} \{0,1\}^m$ and a leakage from K (that does not depend on x^*) of at most ℓ-bit for a certain leakage bound $\ell < \kappa$. Hazay et al. [HLWW16] constructed an LR-wPRF for any polynomial $\ell = \mathsf{poly}(\lambda)$ based solely on the existence of a OWF.

Then, our first (failed) attempt for constructing an HBG from a SNARG and an LR-wPRF is as follows:

- $\mathsf{HBG.Setup}(1^\lambda, 1^k)$ samples $(x_1, ..., x_k) \in \{0,1\}^{m\times k}$ and outputs it as a CRS crs.
- $\mathsf{HBG.GenBits}(\mathsf{crs})$ randomly picks a key $K \xleftarrow{\$} \{0,1\}^\kappa$ of the LR-wPRF, and outputs a commitment com of K by a statistically binding commiement scheme, hidden-bits $r := (F_K(x_1), ..., F_K(x_k))$, and proofs $\{\pi_i\}_{i\in[k]}$ that are generated by the SNARG to certify r.
- $\mathsf{HBG.Verify}(\mathsf{crs}, \mathsf{com}, i, r_i, \pi_i)$ verifies the proof π_i by the verification algorithm of the SNARG.

The binding property easily follows from the statistical binding property of the underlying commitment scheme and the soundness of the underlying SNARG. For the hiding property, we would like to rely on the security of the underlying LR-wPRF by viewing the SNARG proofs as a leakage. However, there are the following two problems:

1. An adversary against the hiding property can obtain all proofs $\{\pi_i\}_{i\in I}$ corresponding to the subset I whose size may be linear in k. On the other hand, for ensuring the succinctness of the commitment, we have to set $k \gg \kappa$. Thus, the total size of $\{\pi_i\}_{i\in I}$ may be larger than κ, in which case it is impossible to rely on the security of the LR-wPRF.
2. Even if the above problem is resolved, we still cannot apply the security of the LR-wPRF since com also depends on K and its size must be larger than that of K.

To resolve these issues, our idea is to drop the commitment com from the output of $\mathsf{HBG.GenBits}(\mathsf{crs})$, and generate a single SNARG proof π that proves that "there exists $K \in \{0,1\}^\kappa$ such that $r_i = F_K(x_i)$ for all $i \in I$" in one-shot instead of generating π_i for each $i \in I$ separately. Then, the only leakage of K given to an adversary against the hiding property is the SNARG proof π, whose size is sublinear in $|I|$ by the succinctness of the SNARG. Thus, it seems possible to apply the security of the LR-wPRF if we set parameters appropriately. However, this idea is not compatible with the syntax of an HBG. This is why we modify the syntax of an HBG to introduce what we call an *HBG with subset-dependent proofs* (SDP-HBG).

HBG with Subset-Dependent Proofs. Roughly speaking, an SDP-HBG is a (weaker) variant of an HBG with the following modifications:

1. A proof is generated depending on a subset I, which specifies positions of bits to be revealed. This is in contrast to the original definition of an HBG where proofs are generated for each position $i \in [k]$. To formalize this, we introduce the proving algorithm separated from the bits generation algorithm.
2. The bits generation algorithm does not output a commitment, and we require a relaxed version of the binding property that we call the *somewhat binding* property as explained later.

More precisely, an SDP-HBG consists of the following algorithms:

- $\mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Setup}(1^\lambda, 1^k)$ generates a CRS crs.
- $\mathsf{HBG}^{\mathsf{sdp}}.\mathsf{GenBits}(\mathsf{crs})$ generates "hidden-bits" $r \in \{0,1\}^k$ and a state st.
- $\mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Prove}(\mathsf{st}, I)$ generates a proof π that certifies the sub-string r_I.
- $\mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Verify}(\mathsf{crs}, I, r_I, \pi)$ verifies the proof π to ensure that the substring of r on the positions corresponding to the subset I is indeed r_I.

We require an SDP-HBG to satisfy the somewhat binding property and the hiding property. The somewhat binding property requires that there exists a "sparse" subset $\mathcal{V}^{\mathsf{crs}} \in \{0,1\}^k$ (dependent on crs) of size much smaller than 2^k such that no PPT malicious prover can generate a proof for bits that are not consistent with any element of $\mathcal{V}^{\mathsf{crs}}$. As mentioned earlier, a similar property easily follows by combining the succinctness of the commitment and the binding property in the original HBG, and this was the essential property to prove soundness in the construction of a NIZK from an HBG. The hiding property is similar to that for an HBG except that an adversary is given a single proof π corresponding to the subset I instead of $\{\pi_i\}_{i\in I}$. Namely, it requires that for any subset $I \in [k]$, no PPT adversary given $\{r_i\}_{i\in I}$ and π that certifies $\{r_i\}_{i\in I}$ can distinguish $r_{\overline{I}}$ from a fresh random string $r'_{\overline{I}} \xleftarrow{\$} \{0,1\}^{|I|}$, where $r_{\overline{I}}$ denotes the sub-string of r on the positions corresponding to $\overline{I} = [k] \setminus I$.

To see that an SDP-HBG is a weaker primitive than an HBG, in Section 4.2, we formally show that an original HBG indeed implies an SDP-HBG.

SDP-HBG from a SNARG and an LR-wPRF. Next, we construct an SDP-HBG from a SNARG and an LR-wPRF. Since the idea is already explained, we directly give the construction below:

- $\mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Setup}(1^\lambda, 1^k)$ samples $(x_1, ..., x_k) \in \{0,1\}^{m\times k}$ and outputs it as a CRS crs.
- $\mathsf{HBG}^{\mathsf{sdp}}.\mathsf{GenBits}(\mathsf{crs})$ randomly picks a key $K \xleftarrow{\$} \{0,1\}^\kappa$ of the LR-wPRF and outputs hidden-bits $r := (F_K(x_1), ..., F_K(x_k))$ and a state $\mathsf{st} := K$.
- $\mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Prove}(\mathsf{st}, I)$ outputs a SNARG proof π that proves that there exists $K \in \{0,1\}^\kappa$ such that $r_i = F_K(x_i)$ for all $i \in I$.
- $\mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Verify}(\mathsf{crs}, I, r_I, \pi)$ verifies the proof π by the verification algorithm of the SNARG.

The somewhat binding property is easy to reduce to the soundness of the underlying SNARG if $\kappa \ll k$. The hiding property is easy to reduce to the security of the underlying LR-PRF if $|\pi| \le \ell$ where ℓ is the leakage bound by noting that the proof π corresponding to the subset I does not depend on $x_{\overline{I}}$, and thus we can think of $x_{\overline{I}}$ as challenge inputs and π as a leakage. Therefore, what remains is to show that we can appropriately set the parameters to satisfy these two inequalities. Here, for simplicity we assume that the SNARG is fully succinct, i.e., $|\pi| = \mathsf{poly}(\lambda)$ independently of the statement/witness size.[6] Especially, $|\pi|$ can be upper bounded by a polynomial in λ that does not depend on k. Then, we first set $\ell = \mathsf{poly}(\lambda)$ so that $|\pi| \le \ell$. According to this choice of ℓ, $\kappa = \mathsf{poly}(\lambda)$ is determined. Here, we emphasize that κ does not depend on k. Thus, for sufficiently large $k = \mathsf{poly}(\lambda)$, we have $\kappa \ll k$ as desired.[7] The crucial

[6] Though the full succinctness just says $|\pi| = \mathsf{poly}(\lambda)(|x| + |w|)^{o(1)}$, this implies $|\pi| = \mathsf{poly}(\lambda)$ as long as we have $|x| = \mathsf{poly}(\lambda)$ and $|w| = \mathsf{poly}(\lambda)$.

[7] It suffices that we have this for sufficiently large k since we can take $k = \mathsf{poly}(\lambda)$ arbitrarily largely in the construction of a NIZK.

point is that no matter how large k is, this does not affect $|\pi|$ thanks to the full succinctness of the SNARG. We note that we assume nothing about the leakage-rate (i.e., ℓ/κ) of the LR-wPRF, and thus we can use the LR-wPRF based on a OWF in [HLWW16], which achieves a relatively poor leakage-rate of $O(\frac{\log \lambda}{\lambda})$. For the case of slightly-succinct SNARGs, a more careful analysis is needed, but we can extend the above proof as long as $|\pi| = \mathsf{poly}(\lambda)(|x|+|w|)^c$ holds for some constant $c < 1/2$.

As seen above, the underlying SNARG in fact needs to prove only a statement of an NP language with a specific form that is dependent on the LR-wPRF (which is in turn based on a OWF). Thus, if the latter is determined beforehand, the SNARG is required to support this particular language (and not all NP languages).[8]

NIZK from an SDP-HBG. Then, we show that an SDP-HBG suffices for constructing a NIZK. In fact, the construction and security proof are essentially the same as that from an HBG in [QRW19]: The setup algorithm generates a CRS $\mathsf{crs} \xleftarrow{\$} \mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Setup}(1^\lambda, 1^k)$ of the SDP-HBG and a random string $s \xleftarrow{\$} \{0,1\}^k$, and outputs them as a CRS of the NIZK; The prover generates $(r, \mathsf{st}) \xleftarrow{\$} \mathsf{HBG}^{\mathsf{sdp}}.\mathsf{GenBits}(\mathsf{crs})$, sets $\rho := r \oplus s$, runs the prover of the underlying HBM-NIZK w.r.t. the hidden-bits ρ to generate (I, π_{hbm}), generates $\pi_{\mathsf{bgen}} \xleftarrow{\$} \mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Prove}(\mathsf{st}, I)$, and outputs $(I, \pi_{\mathsf{hbm}}, r_I, \pi_{\mathsf{bgen}})$ as a proof of the NIZK; The verifier runs the verification of the underlying SDP-HBG to check the validity of r_I and the verification of the underlying HBM-NIZK under the revealed hidden-bits $\rho_I := r_I \oplus s_I$.

It is easy to see that essentially the same proofs as the NIZK from an HBG work for soundness and non-adaptive zero-knowledge. However, it is not clear how to prove the adaptive zero-knowledge for this construction. As mentioned earlier, for the construction of a NIZK from an HBG, Quach et al. [QRW19] proved its adaptive zero-knowledge assuming that the underlying HBM-NIZK satisfies a stronger notion of zero-knowledge called special zero-knowledge. However, their proof does not extend to the proof of adaptive zero-knowledge for the above NIZK from an SDP-HBG even if we rely on the special zero-knowledge for the underlying HBM-NIZK. Roughly speaking, the problem comes from the fact that the SDP-HBG enables us to generate a proof π_{bgen} corresponding to a subset I only after I is fixed. This is in contrast to an HBG where we can generate π_i that certifies the i-th hidden bit for each $i \in [k]$ before I is fixed. Specifically, the proof of adaptive zero-knowledge from an HBG in [QRW19] crucially relies on the fact that if $I \subseteq I^*$, then a set of proofs $\{\pi_i\}_{i \in I}$ can be derived from $\{\pi_i\}_{i \in I^*}$ in a trivial manner. On the other hand, we do not have a similar property in SDP-HBG since it generates a proof for a subset I in one-shot instead of generating a proof in a bit-by-bit manner. Thus, we have to come up with an alternative way to achieve adaptive zero-knowledge.

Non-adaptive to Adaptive Zero-Knowledge. Based on existing works, we give a generic compiler from non-adaptive to adaptive zero-knowledge. First, we observe that we can construct an HBG by combining a commitment, a pseudorandom generator (PRG), and a NIZK in a straightforward manner. We note that essentially the same construction was already mentioned by Dwork and Naor [DN07] where they constructed a verifiable PRG, which is a similar but slightly different primitive from an HBG. Our crucial observation is that non-adaptive zero-knowledge is sufficient for this construction of an HBG. Then, we can apply the construction of [QRW19] instantiated with the above HBG and an HBM-NIZK with special zero-knowledge to obtain a NIZK with adaptive zero-knowledge.

[8] A similar remark applies to the underlying NIZK with non-adaptive zero-knowledge used in the non-adaptive-to-adaptive conversion for a NIZK.

1.3 Related Work

Known Constructions of NIZKs. Here, we review known constructions of a NIZK for all NP languages. Below, we just write NIZK to mean NIZK for all NP languages for simplicity. In this paragraph, we omit a NIZK that is also a SNARG since such schemes are mentioned in the next paragraph. Blum, Feldman, and Micali [BFM88] introduced the concept of NIZK and constructed a NIZK based on the quadratic residuosity assumption. Feige, Lapidot, and Shamir [FLS99] established the hidden-bits paradigm and constructed a NIZK based on trapdoor permutations. The requirements on trapdoor permutations for realizing a NIZK have been revisited and refined in a series of works [Gol11, GR13, CL18]. Canetti, Halevi, and Katz [CHK07] constructed a NIZK based on pairing by instantiating the hidden-bits paradigm. Groth, Ostrovsky, and Sahai [GOS12] constructed a pairing-based NIZK based on a completely different approach, which yields the first NIZK with perfect zero-knowledge. Sahai and Waters [SW14] constructed the first NIZK with a deterministic proving algorithm and perfect zero-knowledge based on indistinguishability obfuscation and a OWF. Recently, there has been a line of researches [KRR17, CCRR18, CCH+19] aiming at realizing the Fiat-Shamir transform [FS87] in the standard model. Peikert and Shiehian [PS19] constructed a NIZK based on a standard lattice-based assumption following this approach. Very recently, Couteau, Katsumata, and Ursu [CKU20] constructed a NIZK based on a certain exponential hardness assumption on pairing-free groups. We note that it still remains open to construct a NIZK from polynomial hardness assumption on pairing-free groups.

We omit NIZKs in a different model than the CRS model including preprocessing, designated prover, and designated verifier models since our focus in this paper is constructions in the CRS model. We refer to [KW18, KNYY19] for a survey on NIZKs in these models.

Known Constructions of SNARGs. Here, we review known constructions of a SNARG for all NP languages. Below, we just write SNARG to mean SNARG for all NP languages. We note that some of the following constructions are actually a *SNARK*, which satisfies a stronger notion of soundness called extractability, but we just call them a SNARG since we do not discuss extractability in this paper. Also, other than [Mic00, CMS19], here we only mention works that do not rely on random oracles. For the recent advances on practical SNARGs (SNARKs) including those in the random oracle model, see, e.g., the recent papers [BFS20, CHM+20, COS20, EFKP20] and references therein.

Micali [Mic00] constructed a zero-knowledge SNARG in the random oracle model. Chiesa, Manohar, and Spooner [CMS19] proved that the Micali's construction is also secure in the quantum random oracle model. Groth [Gro10, Gro16] and Gennaro, Gentry, Perno, and Raykova [GGPR13] proposed zero-knowledge SNARGs in the CRS model based on non-falsifiable assumptions on pairing groups. There are several constructions of (zero-knowledge) SNARGs in the designated-verifier model where verification can be done only by a designated verifier who possesses a secret verification key. These include constructions based on an extractable collision-resistant hash function [BCC+17], homomorphism-extractable encryption [BC12], linear-only encryption [BCI+13, BISW17, BISW18], etc.

NIZKs/SNARGs and OWFs. Pass and shelat [Ps05] showed that a NIZK for a hard-on-average language implies the existence of (non-uniform) OWFs. On the other hand, Wee [Wee05] gave an evidence that a SNARG for a hard-on-average language is unlikely to imply the existence of OWFs. Therefore, it is considered reasonable to additionally assume the existence of OWFs for constructing a NIZK from a SNARG.

2 Preliminaries

In this section, we review the basic notation and definitions of cryptographic primitives.

Basic Notation. For a natural number $n > 0$, we define $[n] := \{1, \ldots, n\}$. Furthermore, for $I \subseteq [n]$, we define $\overline{I} := [n] \setminus I$.

For a string x, $|x|$ denotes the bit-length of x. For bits $b, b' \in \{0,1\}$, $(b' \stackrel{?}{=} b)$ is defined to be 1 if $b' = b$ holds and 0 otherwise.

For a set S, $|S|$ denotes its size, and $x \xleftarrow{\$} S$ denotes sampling x uniformly at random from S. Furthermore, for natural numbers i, k such that $i \in [k]$ and a sequence $z \in S^k$, z_i denotes the i-th entry in z. Also, for $I \subseteq [k]$, we define $z_I := (z_i)_{i \in I}$, namely the subsequence of z in the positions I.

For a probabilistic (resp. deterministic) algorithm A, $y \xleftarrow{\$} \mathsf{A}(x)$ (resp. $y \leftarrow \mathsf{A}(x)$) denotes $\mathcal{A}$ on input x outputs y. If we need to specify a randomness r used in A, we write $y \leftarrow \mathsf{A}(x; r)$ (in which case the computation is deterministic). If $\mathcal{O}$ is a function or an algorithm, then $\mathsf{A}^{\mathcal{O}}$ means that A has oracle access to $\mathcal{O}$.

Throughout the paper, we use λ to denote the security parameter, and a "PPT adversary" is a non-uniform PPT adversary (equivalently, a family of polynomial-sized circuits). A function $\epsilon(\lambda)$ with range $[0,1]$ is said to be negligible if $\epsilon(\lambda) = \lambda^{-\omega(1)}$, and $\mathsf{negl}(\lambda)$ denotes an unspecified negligible function of λ. $\mathsf{poly}(\lambda)$ denotes an unspecified (constant-degree) polynomial of λ.

2.1 PRG

Definition 1 (Pseudorandom Generator)**.** *Let* $\mathsf{PRG} : \{0,1\}^\lambda \to \{0,1\}^k$ *be an efficiently computable function where* $k = k(\lambda) > \lambda$ *is some polynomial. We say that* PRG *is a* pseudorandom generator (PRG) *if for all PPT adversaries* $\mathcal{A}$*, we have*

$$\left|\Pr[\mathcal{A}(1^\lambda, \mathsf{PRG}(s)) = 1 : s \xleftarrow{\$} \{0,1\}^\lambda] - \Pr[\mathcal{A}(1^\lambda, r) = 1 : r \xleftarrow{\$} \{0,1\}^k]\right| = \mathsf{negl}(\lambda).$$

It is known that a PRG exists if and only if a OWF exists [HILL99].

2.2 Non-interactive Commitment

Here, we review the definition of a statistically binding non-interactive commitment scheme. We will use one that uses a CRS, which is sufficient for our purpose.

Definition 2 (Non-interactive Commitment)**.** *A* statistically binding non-interactive commitment *scheme (in the CRS model) consists of the two algorithms* $(\mathsf{Com.Setup}, \mathsf{Com.Commit})$*:*

$\mathsf{Com.Setup}(1^\lambda) \xrightarrow{\$} \mathsf{crs}$*: The setup algorithm takes the security parameter* 1^λ *as input, and outputs a CRS* crs*.*

$\mathsf{Com.Commit}(\mathsf{crs}, m) \xrightarrow{\$} \mathsf{com}$*: The commitment generation algorithm takes a CRS* crs *and a message* m *as input, and outputs a commitment* com*.*

We require the statistical binding and computational hiding properties defined below:

Statistical Binding: *We have*

$$\Pr\left[\begin{array}{l} \exists (m, m', r, r') \ s.t. \\ \mathsf{Com.Commit}(\mathsf{crs}, m; r) = \mathsf{Com.Commit}(\mathsf{crs}, m'; r') \\ \wedge\ m \neq m' \end{array} : \mathsf{crs} \xleftarrow{\$} \mathsf{Com.Setup}(1^\lambda)\right] = \mathsf{negl}(\lambda).$$

Computational Hiding: *For all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, we have*

$$\left| \Pr \left[\mathcal{A}_2(\mathsf{com}^*, \mathsf{st}) = b : \begin{array}{l} \mathsf{crs} \xleftarrow{\$} \mathsf{Com.Setup}(1^\lambda); \\ (m_0, m_1, \mathsf{st}) \xleftarrow{\$} \mathcal{A}_1(\mathsf{crs}); \\ b \xleftarrow{\$} \{0,1\}; \\ \mathsf{com}^* \xleftarrow{\$} \mathsf{Com.Commit}(\mathsf{crs}, m_b) \end{array} \right] - \frac{1}{2} \right| = \mathsf{negl}(\lambda).$$

It is known that a statistically binding non-interactive commitment in the CRS model exists if and only if a OWF exists [Nao91, HILL99].

2.3 Public Key Encryption

Here, we review the definition of public key encryption (PKE).

Definition 3 (Public Key Encryption)**.** *A* public key encryption (PKE) *scheme consists of the three algorithms* $(\mathsf{PKE.KeyGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$*:*

$\mathsf{PKE.KeyGen}(1^\lambda) \xrightarrow{\$} (\mathsf{pk}, \mathsf{sk})$*: The key generation algorithm takes the security parameter 1^λ as input, and outputs a public key* pk *and a secret key* sk*.*

$\mathsf{PKE.Enc}(\mathsf{pk}, m) \xrightarrow{\$} \mathsf{ct}$*: The encryption algorithm takes a public key* pk *and a message m as input, and outputs a ciphertext* ct*.*

$\mathsf{PKE.Dec}(\mathsf{sk}, \mathsf{ct}) \to m$*: The decryption algorithm takes a secret key* sk *and a ciphertext* ct *as input, and outputs a message m.*

We require the following properties:

Correctness: *For all messages m, we have*

$$\Pr \left[\mathsf{PKE.Dec}(\mathsf{sk}, \mathsf{ct}) = m : \begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{PKE.KeyGen}(1^\lambda); \\ \mathsf{ct} \xleftarrow{\$} \mathsf{PKE.Enc}(\mathsf{pk}, m) \end{array} \right] = 1.$$

CPA Security: *For all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, we have*

$$\left| \Pr \left[\mathcal{A}_2(\mathsf{ct}^*, \mathsf{st}) = b : \begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{PKE.KeyGen}(1^\lambda); \\ (m_0, m_1, \mathsf{st}) \xleftarrow{\$} \mathcal{A}_1(\mathsf{pk}); \\ b \xleftarrow{\$} \{0,1\}; \\ \mathsf{ct}^* \xleftarrow{\$} \mathsf{PKE.Enc}(\mathsf{pk}, m_b) \end{array} \right] - \frac{1}{2} \right| = \mathsf{negl}(\lambda).$$

2.4 NIZK and SNARG

Here, we define several notions of a non-interactive argument for an NP language $\mathcal{L}$. Throughout this paper, for an NP language $\mathcal{L}$, we denote by $\mathcal{R} \subseteq \{0,1\}^* \times \{0,1\}^*$ the corresponding efficiently computable binary relation. For $(x, w) \in \mathcal{R}$, we call x a statement and w a witness.

Definition 4 (Non-interactive Arguments)**.** *A non-interactive argument for an* NP *language $\mathcal{L}$ consists of the three PPT algorithms* $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$*:*

$\mathsf{Setup}(1^\lambda) \xrightarrow{\$} \mathsf{crs}$*: The setup algorithm takes the security parameter 1^λ as input, and outputs a CRS* crs*.*

$\mathsf{Prove}(\mathsf{crs}, x, w) \xrightarrow{\$} \pi$: *The prover's algorithm takes a CRS* crs*, a statement* x*, and a witness* w *as input, and outputs a proof* π*.*

$\mathsf{Verify}(\mathsf{crs}, x, \pi) \to \top$ *or* $\bot$*: The verifier's algorithm takes a CRS* crs*, a statement* x*, and a proof* π *as input, and outputs* $\top$ *to indicate acceptance of the proof or* $\bot$ *otherwise.*

A non-interactive argument must satisfy the following requirements:

Completeness: *For all pairs* $(x, w) \in \mathcal{R}$*, we have*

$$\Pr\left[\mathsf{Verify}(\mathsf{crs}, x, \pi) = \top : \begin{array}{l} \mathsf{crs} \xleftarrow{\$} \mathsf{Setup}(1^\lambda); \\ \pi \xleftarrow{\$} \mathsf{Prove}(\mathsf{crs}, x, w) \end{array}\right] = 1.$$

Soundness: *We define the following four variants of soundness.*

Adaptive Computational Soundness: *For all PPT adversaries* $\mathcal{A}$*, we have*

$$\Pr\left[x \notin \mathcal{L} \wedge \mathsf{Verify}(\mathsf{crs}, x, \pi) = \top : \begin{array}{l} \mathsf{crs} \xleftarrow{\$} \mathsf{Setup}(1^\lambda); \\ (x, \pi) \xleftarrow{\$} \mathcal{A}(\mathsf{crs}) \end{array}\right] = \mathsf{negl}(\lambda).$$

Adaptive Statistical Soundness: *This is defined similarly to adaptive computational soundness, except that* $\mathcal{A}$ *can be any computationally unbounded adversary.*

Non-adaptive Computational (resp. Statistical) Soundness: *This is defined similarly to adaptive computational (resp. statistical) soundness, except that* $\mathcal{A}$ *must declare* $x \notin \mathcal{L}$ *before it is given* crs*.*

If we only require completeness and soundness as defined above, a non-interactive argument trivially exists for all NP languages, since a witness itself can be used as a proof. Thus, we consider two other properties that make non-interactive arguments non-trivial. First, we define non-interactive zero-knowledge arguments (NIZKs).

Definition 5 (NIZK)**.** *A non-interactive argument* $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ *for an* NP *language* $\mathcal{L}$ *is a* non-interactive zero-knowledge argument *(NIZK) if it satisfies the following property in addition to completeness and soundness.*

(Computational) Zero-Knowledge: *We define the following four variants of zero-knowledge property.*

Adaptive Multi-theorem Zero-Knowledge: *There exists a PPT simulator* $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ *that satisfies the following. For all PPT adversaries* $\mathcal{A}$*, we have*

$$\left|\Pr[\mathsf{Expt}^{\mathsf{azk\text{-}real}}_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Expt}^{\mathsf{azk\text{-}sim}}_{\mathcal{A},\mathcal{S}}(\lambda) = 1]\right| = \mathsf{negl}(\lambda),$$

where the experiments $\mathsf{Expt}^{\mathsf{azk\text{-}real}}_{\mathcal{A}}(\lambda)$ *and* $\mathsf{Expt}^{\mathsf{azk\text{-}sim}}_{\mathcal{A},\mathcal{S}}(\lambda)$ *are defined as follows, and in the experiments,* $\mathcal{A}$*'s queries* (x, w) *must satisfy* $(x, w) \in \mathcal{R}$*.*

$\underline{\mathsf{Expt}^{\mathsf{azk\text{-}real}}_{\mathcal{A}}(\lambda):}$
$\mathsf{crs} \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$
$b' \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_0(\cdot,\cdot)}(\mathsf{crs})$
where $\mathcal{O}_0(x, w)$ *returns* $\mathsf{Prove}(\mathsf{crs}, x, w)$
Return b'.

$\underline{\mathsf{Expt}^{\mathsf{azk\text{-}sim}}_{\mathcal{A},\mathcal{S}}(\lambda):}$
$(\mathsf{crs}, \mathsf{st}) \xleftarrow{\$} \mathcal{S}_1(1^\lambda)$
$b' \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_1(\cdot,\cdot)}(\mathsf{crs})$
where $\mathcal{O}_1(x, w)$ *returns* $\mathcal{S}_2(\mathsf{st}, x)$
Return b'.

Though we treat adaptive multi-theorem zero-knowledge as defined above as a default notion of zero-knowledge, we also define weaker notions of zero-knowledge.

Adaptive Single-Theorem Zero-Knowledge: *This is defined similarly to adaptive multi-theorem zero-knowledge, except that $\mathcal{A}$ is allowed to make only a single query.*

Non-adaptive Multi-theorem Zero-Knowledge: *There exists a PPT simulator $\mathcal{S}$ that satisfies the following. For all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ we have*

$$\left|\Pr[\mathsf{Expt}^{\mathsf{nazk\text{-}real}}_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Expt}^{\mathsf{nazk\text{-}sim}}_{\mathcal{A},\mathcal{S}}(\lambda) = 1]\right| = \mathsf{negl}(\lambda),$$

where the experiments $\mathsf{Expt}^{\mathsf{nazk\text{-}real}}_{\mathcal{A}}(\lambda)$ and $\mathsf{Expt}^{\mathsf{nazk\text{-}sim}}_{\mathcal{A},\mathcal{S}}(\lambda)$ are defined below. In the experiments, ℓ (the number of statement/witness pairs) is arbitrarily chosen by $\mathcal{A}_1$, and $\mathcal{A}_1$'s output must satisfy $(x_i, w_i) \in \mathcal{R}$ for all $i \in [\ell]$.

$\underline{\mathsf{Expt}^{\mathsf{nazk\text{-}real}}_{\mathcal{A}}(\lambda):}$
$(\{(x_i, w_i)\}_{i\in[\ell]}, \mathsf{st}) \xleftarrow{\$} \mathcal{A}_1(1^\lambda)$
$\mathsf{crs} \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$
$\pi_i \xleftarrow{\$} \mathsf{Prove}(\mathsf{crs}, x_i, w_i)$ *for* $i \in [\ell]$
$b' \xleftarrow{\$} \mathcal{A}_2(\mathsf{crs}, \{\pi_i\}_{i\in[\ell]}, \mathsf{st})$
Return b'.

$\underline{\mathsf{Expt}^{\mathsf{nazk\text{-}sim}}_{\mathcal{A},\mathcal{S}}(\lambda):}$
$(\{(x_i, w_i)\}_{i\in[\ell]}, \mathsf{st}) \xleftarrow{\$} \mathcal{A}_1(1^\lambda)$
$(\mathsf{crs}, \{\pi_i\}_{i\in[\ell]}) \xleftarrow{\$} \mathcal{S}(1^\lambda, \{x_i\}_{i\in[\ell]})$
$b' \xleftarrow{\$} \mathcal{A}_2(\mathsf{crs}, \{\pi_i\}_{i\in[\ell]}, \mathsf{st})$
Return b'.

Non-adaptive Single-Theorem Zero-Knowledge: *This is defined similarly to non-adaptive multi-theorem zero-knowledge, except that ℓ must be* 1.

It is well-known that a NIZK with adaptive single-theorem zero-knowledge can be generically converted into a NIZK with adaptive multi-theorem zero-knowledge using a PRG [FLS99]. It is easy to see that the same construction works in the non-adaptive setting. Thus, we have the following lemma.

Lemma 1. *If there exist a OWF and a NIZK for all* NP *languages with adaptive (resp. non-adaptive) single-theorem zero-knowledge, then there exists a NIZK for all* NP *languages with adaptive (resp. non-adaptive) multi-theorem zero-knowledge. The resulting NIZK satisfies the same notion of soundness (which is either of adaptive/non-adaptive statistical/computational soundness) as the building-block NIZK.*

Remark 1. *Pass and shelat [Ps05] showed that a NIZK for a hard-on-average language implies the existence of a (non-uniform) OWF. Therefore, we can weaken the assumption of the existence of a OWF to the existence of a hard-on-average* NP *language. We just assume the existence of a OWF for simplicity. A similar remark also applies to Theorem 2 and Lemmata 3 and 4.*

Next, we define SNARGs. The following definition is taken from [GW11] with a minor modification in the definition of slight succinctness (see Remark 2).

Definition 6 ((Fully/Slightly Succinct) SNARG)**.** *A non-interactive argument* $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ *for an* NP *language $\mathcal{L}$ is a fully (resp. δ-slightly)* succinct non-interactive argument *(SNARG) if it satisfies full (resp. δ-slight) succinctness defined as follows in addition to completeness and soundness.*

Succinctness: *We define the following two variants of succinctness.*

Full Succinctness: *For all $(x, w) \in \mathcal{R}$, $\mathsf{crs} \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$, and $\pi \xleftarrow{\$} \mathsf{Prove}(\mathsf{crs}, x, w)$, we have $|\pi| = \mathsf{poly}(\lambda)(|x| + |w|)^{o(1)}$.*

δ-**Slight Succinctness:** *For all* $(x, w) \in \mathcal{R}$, $\mathsf{crs} \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$, *and* $\pi \xleftarrow{\$} \mathsf{Prove}(\mathsf{crs}, x, w)$, *we have* $|\pi| = \mathsf{poly}(\lambda)(|x| + |w|)^\delta$.

Remark 2. *The notion of δ-slight succinctness is meaningful only when $\delta < 1$ since otherwise we can use a witness itself as a proof. We note that our definition of δ-slight succinctness for any $\delta < 1$ is stronger than slight succinctness defined in [GW11] where they require $|\pi| = \mathsf{poly}(\lambda)(|x| + |w|)^\delta + o(|x| + |w|)$ for some $\delta < 1$. Namely, they allow the proof size to grow according to any function dominated by $|x| + |w|$ asymptotically as long as that is independent of the security parameter λ.*

We define an additional property for SNARG.

Definition 7 (Efficient Verification of SNARG)**.** *A SNARG* $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ *for an* NP *language* $\mathcal{L}$ *has efficient verification if the following is satisfied.*

Efficient Verification: *For all* $(x, w) \in \mathcal{R}$, $\mathsf{crs} \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$, *and* $\pi \xleftarrow{\$} \mathsf{Prove}(\mathsf{crs}, x, w)$, *the running time of* $\mathsf{Verify}(\mathsf{crs}, x, \pi)$ *is* $\mathsf{poly}(\lambda)(|x| + |w|)^{o(1)}$.

Remark 3. *The efficient verification property immediately implies full succinctness.*

Remark 4. *The efficient verification property is usually a default requirement for SNARGs. On the other hand, we do not assume a SNARG to have efficient verification unless otherwise mentioned. This is because efficient verification is not needed for the construction of a NIZK in this paper.*

2.5 NIZK in the Hidden-Bits Model

Here, we define a NIZK in the hidden-bits model introduced in [FLS99]. The following definition is taken from [QRW19].

Definition 8 (NIZK in the Hidden-Bits Model)**.** *Let $\mathcal{L}$ be an* NP *language and $\mathcal{R}$ be its associated relation. A* non-interactive zero-knowledge proof in the hidden-bits model (HBM-NIZK) *for $\mathcal{L}$ consists of the pair of PPT algorithms* $(\mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Prove}, \mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Verify})$ *and a polynomial $k = k(\lambda, n)$, which specifies the hidden-bits length.*

$\mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Prove}(1^\lambda, \rho, x, w) \xrightarrow{\$} (I, \pi)$*: The prover's algorithm takes the security parameter 1^λ, a string $\rho \in \{0,1\}^{k(\lambda,n)}$, a statement $x \in \{0,1\}^n$, and a witness w as input, and outputs a subset of indices $I \subseteq [k]$ and a proof π.*

$\mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Verify}(1^\lambda, I, \rho_I, x, \pi) \to \top$ *or* $\bot$*: The verifier's algorithm takes the security parameter 1^λ, a subset $I \subseteq [k]$, a string ρ_I, a statement x, and a proof π as input, and outputs $\top$ to indicate acceptance of the proof or $\bot$ otherwise.*

An HBM-NIZK must satisfy the following requirements.

Completeness: *For all pairs $(x, w) \in \mathcal{R}$, we have*

$$\Pr\left[\mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Verify}(1^\lambda, I, \rho_I, x, \pi) = \top : \begin{array}{l} \rho \xleftarrow{\$} \{0,1\}^{k(\lambda,|x|)}; \\ (I, \pi) \xleftarrow{\$} \mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Prove}(1^\lambda, \rho, x, w) \end{array}\right] = 1.$$

ϵ-**Soundness:** *For all polynomials $n = n(\lambda)$ and computationally unbounded adversaries $\mathcal{A}$, we have*

$$\Pr\left[x \in \{0,1\}^n \setminus \mathcal{L} \wedge \mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Verify}(1^\lambda, I, \rho_I, x, \pi) = \top : \begin{array}{l} \rho \xleftarrow{\$} \{0,1\}^{k(\lambda,n)}; \\ (x, I, \pi) \xleftarrow{\$} \mathcal{A}(\rho) \end{array}\right] \leq \epsilon(\lambda).$$

Zero-Knowledge: *There exists a PPT simulator* $\mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Sim}$ *that satisfies the following. For all computationally unbounded adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, we have*

$$\left|\Pr[\mathsf{Expt}^{\mathsf{hbmzk\text{-}real}}_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Expt}^{\mathsf{hbmzk\text{-}sim}}_{\mathcal{A},\mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Sim}}(\lambda) = 1]\right| = \mathsf{negl}(\lambda),$$

where the experiments $\mathsf{Expt}^{\mathsf{hbmzk\text{-}real}}_{\mathcal{A}}(\lambda)$ *and* $\mathsf{Expt}^{\mathsf{hbmzk\text{-}sim}}_{\mathcal{A},\mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Sim}}(\lambda)$ *are defined as follows, and* $\mathcal{A}_1$*'s output must satisfy* $(x, w) \in \mathcal{R}$*.*

$\underline{\mathsf{Expt}^{\mathsf{hbmzk\text{-}real}}_{\mathcal{A}}(\lambda):}$
$(x, w, \mathsf{st}) \xleftarrow{\$} \mathcal{A}_1(1^\lambda)$
$\rho \xleftarrow{\$} \{0,1\}^{k(\lambda,|x|)}$
$(I, \pi) \xleftarrow{\$} \mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Prove}(1^\lambda, \rho, x, w)$
$b' \xleftarrow{\$} \mathcal{A}_2(I, \rho_I, \pi, \mathsf{st})$
Return b'.

$\underline{\mathsf{Expt}^{\mathsf{hbmzk\text{-}sim}}_{\mathcal{A},\mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Sim}}(\lambda):}$
$(x, w, \mathsf{st}) \xleftarrow{\$} \mathcal{A}_1(1^\lambda)$
$(I, \rho_I, \pi) \xleftarrow{\$} \mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Sim}(1^\lambda, x)$
$b' \xleftarrow{\$} \mathcal{A}_2(I, \rho_I, \pi, \mathsf{st})$
Return b'.

Quach et al. [QRW19] defined a stronger variant of zero-knowledge called special zero-knowledge *as follows.*

Special Zero-Knowledge: *There exists a PPT simulator* $\mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Sim} = (\mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Sim}_1, \mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Sim}_2)$ *such that the following properties hold:*

1. *For any* $(x, w) \in \mathcal{R}$ *and* $\rho \in \{0,1\}^{k(\lambda,|x|)}$*, if we let* $\rho' \xleftarrow{\$} \mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Sim}_1(\rho)$ *and* $(I, \pi) \xleftarrow{\$} \mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Prove}(1^\lambda, \rho, x, w)$*, then* $\rho'_I = \rho_I$*.*
2. *For all computationally unbounded adversaries* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *and all polynomials* $n = n(\lambda)$*, we have*

$$\left|\Pr[\mathsf{Expt}^{\mathsf{hbmszk\text{-}real}}_{\mathcal{A},\mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Sim}}(\lambda) = 1] - \Pr[\mathsf{Expt}^{\mathsf{hbmszk\text{-}sim}}_{\mathcal{A},\mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Sim}}(\lambda) = 1]\right| = \mathsf{negl}(\lambda),$$

where the experiments $\mathsf{Expt}^{\mathsf{hbmszk\text{-}real}}_{\mathcal{A},\mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Sim}}(\lambda)$ *and* $\mathsf{Expt}^{\mathsf{hbmszk\text{-}sim}}_{\mathcal{A},\mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Sim}}(\lambda)$ *are defined as follows, and* $\mathcal{A}_1$*'s output must satisfy* $(x, w) \in \mathcal{R}$ *and* $x \in \{0,1\}^n$*.*

$\underline{\mathsf{Expt}^{\mathsf{hbmszk\text{-}real}}_{\mathcal{A},\mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Sim}}(\lambda):}$
$\rho \xleftarrow{\$} \{0,1\}^{k(\lambda,n)}$
$\rho' \xleftarrow{\$} \mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Sim}_1(\rho)$
$(x, w, \mathsf{st}) \xleftarrow{\$} \mathcal{A}_1(\rho')$
$(I, \pi) \xleftarrow{\$} \mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Prove}(1^\lambda, \rho, x, w)$
$b' \xleftarrow{\$} \mathcal{A}_2(I, \pi, \mathsf{st})$
Return b'.

$\underline{\mathsf{Expt}^{\mathsf{hbmszk\text{-}sim}}_{\mathcal{A},\mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Sim}}(\lambda):}$
$\rho \xleftarrow{\$} \{0,1\}^{k(\lambda,n)}$
$\rho' \xleftarrow{\$} \mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Sim}_1(\rho)$
$(x, w, \mathsf{st}) \xleftarrow{\$} \mathcal{A}_1(\rho')$
$(I, \pi) \xleftarrow{\$} \mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Sim}_2(\rho, x)$
$b' \xleftarrow{\$} \mathcal{A}_2(I, \pi, \mathsf{st})$
Return b'.

Lemma 2 ([QRW19])**.** *For any* NP *language* $\mathcal{L}$*, there exists an HBM-NIZK satisfying completeness,* $2^{-\Omega(k)}$*-soundness, zero-knowledge, and special zero-knowledge.*

Remark 5. *As mentioned in [QRW19], it is easy to see that special zero-knowledge implies zero-knowledge. We state them separately for convenience in subsequent sections. Specifically, we use zero-knowledge in Section 5 where we construct a NIZK with non-adaptive zero-knowledge from a primitive that we call an* HBG with subset-dependent proofs (SDP-HBG) *(see Section 4.1 for its formal definition), and special zero-knowledge in Appendix A.2 where we construct a NIZK with adaptive zero-knowledge from a HBG.*

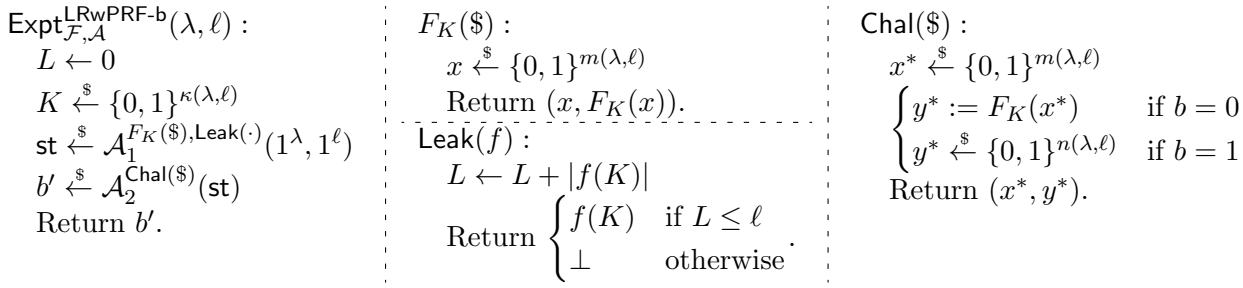

Figure 1: The experiment for defining the leakage-resilience for a wPRF.

2.6 Leakage-Resilient Weak Pseudorandom Function

Here, we review the definition of a leakage-resilient weak pseudorandom function (LR-wPRF) [HLWW16]. Though the definition is essentially the same as that in [HLWW16], we make it more explicit that we can arbitrarily set the leakage bound $\ell = \ell(\lambda)$ instead of treating ℓ as a fixed parameter hardwired in a scheme.[9] Specifically, we define parameters of an LR-wPRF including the key length, input length, and output length as polynomials of λ and ℓ. This implicitly means that an evaluation of an LR-wPRF also depends on ℓ since it is given a key and an input whose length depends on ℓ.

Definition 9 (Leakage-Resilient Weak Pseudorandom Function)**.** *Let $\kappa = \kappa(\lambda,\ell)$, $m = m(\lambda,\ell)$, and $n = n(\lambda,\ell)$ be polynomials. A* leakage-resilient weak PRF (LR-wPRF) *with the key length κ, input length m, and output length n, is a family of efficiently computable functions $\mathcal{F} = \{F_K : \{0,1\}^m \to \{0,1\}^n\}_{K\in\{0,1\}^\kappa}$ such that for all polynomials $\ell = \ell(\lambda)$ and PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, we have*

$$\left|\Pr[\mathsf{Expt}^{\mathsf{LRwPRF\text{-}0}}_{\mathcal{F},\mathcal{A}}(\lambda,\ell) = 1] - \Pr[\mathsf{Expt}^{\mathsf{LRwPRF\text{-}1}}_{\mathcal{F},\mathcal{A}}(\lambda,\ell) = 1]\right| = \mathsf{negl}(\lambda),$$

where the experiment $\mathsf{Expt}^{\mathsf{LRwPRF\text{-}b}}_{\mathcal{F},\mathcal{A}}(\lambda,\ell)$ (with $b \in \{0,1\}$) is described in Figure 1.

Hazay et al. [HLWW16] showed how to construct an LR-wPRF from a OWF. Their result can be stated in the following form that is convenient for our purpose.

Theorem 1 ([HLWW16])**.** *If there exists a OWF, then there exists an LR-wPRF with the key length $\kappa = \ell \cdot \mathsf{poly}(\lambda)$, input length $m = \ell \cdot \mathsf{poly}(\lambda)$, and output length $n = 1$.*

Remark 6. *Actually, Hazay et al. showed that we can set $\kappa = O(\ell\lambda/\log\lambda)$, $m = O(\ell\lambda)$, and n to be any polynomial in λ. We state the theorem in the above form since this is sufficient for our purpose.*

2.7 Hidden-Bits Generator

A hidden-bits generator (HBG) is a primitive proposed by Quach, Rothblum, and Wichs [QRW19]. Our definition here is slightly different from the original one since we require a slightly stronger requirement of the succinct commitment property and we define both computational and statistical binding whereas the original definition only considered statistical one.[10]

[9] Syntactically, this treatment of the leakage bound ℓ is similar to a cryptographic primitive in the bounded retrieval model (BRM). Unlike the BRM, however, we do not pose any efficiency requirement on the scheme regarding the dependency on the given leakage bound ℓ.

[10] We note that Libert et al. [LPWW20] also introduced a computational binding property for an HBG, but their definition is slightly different from ours since they focus on an HBG that satisfies a stronger form of hiding property called statistical simulation property.

Definition 10 (Hidden-Bits Generator [QRW19])**.** *A hidden-bits generator (HBG) consists of the three PPT algorithms* (HBG.Setup, HBG.GenBits, HBG.Verify)*:*

$\mathsf{HBG.Setup}(1^\lambda, 1^k) \xrightarrow{\$} \mathsf{crs}$*: The setup algorithm takes the security parameter* 1^λ *and the length parameter* 1^k *as input, and outputs a CRS* crs*.*

$\mathsf{HBG.GenBits}(\mathsf{crs}) \xrightarrow{\$} (\mathsf{com}, r, \{\pi_i\}_{i\in[k]})$*: The bits generation algorithm takes a CRS* crs *as input, and outputs a commitment* com*, a string* $r \in \{0,1\}^k$*, and a tuple of proofs* $\{\pi_i\}_{i\in[k]}$*.*

$\mathsf{HBG.Verify}(\mathsf{crs}, \mathsf{com}, i, r_i, \pi_i) \to \top$ *or* $\bot$*: The verification algorithm takes a CRS* crs*, a commitment* com*, a position* $i \in [k]$*, a bit* $r_i \in \{0,1\}$*, and a proof* π_i *as input, and outputs* $\top$ *indicating acceptance or* $\bot$ *otherwise.*

We require an HBG to satisfy the following properties:

Correctness: *For all natural numbers* k *and* $i \in [k]$*, we have*

$$\Pr\left[\mathsf{HBG.Verify}(\mathsf{crs}, \mathsf{com}, i, r_i, \pi_i) = \top : \begin{array}{l} \mathsf{crs} \xleftarrow{\$} \mathsf{HBG.Setup}(1^\lambda, 1^k); \\ (\mathsf{com}, r, \{\pi_i\}_{i\in[k]}) \xleftarrow{\$} \mathsf{HBG.GenBits}(\mathsf{crs}) \end{array}\right] = 1.$$

Succinct Commitment: *There exists a constant* $\gamma < 1$ *such that the bit-length of a commitment* com *is at most* $k^\gamma \mathsf{poly}(\lambda)$ *where* $\mathsf{crs} \xleftarrow{\$} \mathsf{HBG.Setup}(1^\lambda, 1^k)$ *and* $(\mathsf{com}, r, \{\pi_i\}_{i\in[k]}) \xleftarrow{\$} \mathsf{HBG.GenBits}(\mathsf{crs})$*.*[11]

Computational (resp. Statistical) Binding: *There exists a possibly inefficient deterministic algorithm* Open *such that for any polynomial* $k = k(\lambda)$ *and PPT (resp. computationally unbounded) adversary* $\mathcal{A}$*, we have*

$$\Pr\left[r_i^* \neq r_i \wedge \mathsf{HBG.Verify}(\mathsf{crs}, \mathsf{com}, i, r_i^*, \pi_i) = \top : \begin{array}{l} \mathsf{crs} \xleftarrow{\$} \mathsf{HBG.Setup}(1^\lambda, 1^k); \\ (\mathsf{com}, i, r_i^*, \pi_i) \xleftarrow{\$} \mathcal{A}(\mathsf{crs}); \\ r \leftarrow \mathsf{Open}(1^k, \mathsf{crs}, \mathsf{com}) \end{array}\right] = \mathsf{negl}(\lambda).$$

Computational Hiding: *For any polynomial* $k = k(\lambda)$*,* $I \subseteq [k]$*, and PPT adversary* $\mathcal{A}$*, we have*

$$\left|\Pr[\mathcal{A}(\mathsf{crs}, I, \mathsf{com}, r_I, \pi_I, r_{\overline{I}}) = 1] - \Pr[\mathcal{A}_2(\mathsf{crs}, I, \mathsf{com}, r_I, \pi_I, r'_{\overline{I}}) = 1]\right| = \mathsf{negl}(\lambda),$$

where $\mathsf{crs} \xleftarrow{\$} \mathsf{HBG.Setup}(1^\lambda, 1^k)$*,* $(\mathsf{com}, r, \{\pi_i\}_{i\in[k]}) \xleftarrow{\$} \mathsf{HBG.GenBits}(\mathsf{crs})$*, and* $r' \xleftarrow{\$} \{0,1\}^k$*.*

3 Non-Adaptive to Adaptive Zero-Knowledge for NIZK

In this section, we show the following theorem.

Theorem 2. *If there exist a OWF and a NIZK for all* NP *languages that satisfies adaptive computational (resp. statistical) soundness and non-adaptive single-theorem zero-knowledge, then there exists a NIZK for all* NP *languages that satisfies adaptive computational (resp. statistical) soundness and adaptive multi-theorem zero-knowledge.*

[11] We note that the requirement of the succinct commitment property given here is slightly stronger than that in the original definition in [QRW19]. We use this definition since it is simpler and our construction satisfies this property.

Remark 7. *The theorem remains true even if we start from a NIZK with non-adaptive statistical soundness since we can convert it into one with adaptive statistical soundness while preserving the zero-knowledge property by a simple parallel repetition. On the other hand, we do not know whether the theorem remains true if we start from a NIZK with non-adaptive computational soundness.*

HBG from Non-adaptive NIZK. First, we show that we can construct an HBG by combining a non-interactive commitment scheme, a PRG, and a NIZK in a straightforward manner. We note that Dwork and Naor [DN07] already mentioned a similar construction.[12] Our crucial observation is that non-adaptive multi-theorem zero-knowledge is sufficient for this purpose. Moreover, as stated in Lemma 1, we can generically upgrade non-adaptive single-theorem zero-knowledge to non-adaptive multi-theorem zero-knowledge. Therefore, we obtain the following lemma.

Lemma 3. *If there exist a OWF and a NIZK for all* NP *languages that satisfies adaptive computational (resp. statistical) soundness and non-adaptive single-theorem zero-knowledge, then there exists an HBG that satisfies succinct commitment, computational (resp. statistical) binding, and computational hiding.*

Since the construction and security proof are straightforward, we give them in Appendix A.1.

Adaptive NIZK from HBG. Quach et al. [QRW19] gave a construction of a NIZK with adaptive statistical soundness and adaptive multi-theorem zero-knowledge based on an HBG with statistical binding and computational hiding. It is easy to see that the same construction works for a computationally binding HBG to construct an adaptively computationally sound NIZK. Namely, we have the following lemma.

Lemma 4. *If there exist a OWF and an HBG that satisfies succinct commitment, computational (resp. statistical) binding, and computational hiding, then there exists a NIZK for all* NP *languages that satisfies adaptive computational (resp. statistical) soundness and adaptive multi-theorem zero-knowledge.*

Though the construction and proof are essentially the same as those in [QRW19], we give them in Appendix A.2 for completeness.

Theorem 2 can be obtained by combining Lemmata 3 and 4.

4 Hidden-Bits Generator with Subset-Dependent Proofs

In this section, we introduce a weaker variant of an HBG that we call an *HBG with subset-dependent proofs* (SDP-HBG). We also give a construction of an SDP-HBG from the combination of a SNARG and an LR-wPRF (and thus, from a SNARG and a OWF).

4.1 Definition

Here, we define an SDP-HBG.

Definition 11 (SDP-HBG)**.** *A* hidden-bits generator with subset dependent proofs (SDP-HBG) *consists of the four PPT algorithms* $(\mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Setup}, \mathsf{HBG}^{\mathsf{sdp}}.\mathsf{GenBits}, \mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Prove}, \mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Verify})$*:*

[12] Dwork and Naor [DN07] constructed what they call a *verifiable pseudorandom generator* from a NIZK, which is a similar primitive to an HBG.

$\mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Setup}(1^\lambda, 1^k) \xrightarrow{\$} \mathsf{crs}$: *The setup algorithm takes the security parameter* 1^λ *and the length parameter* 1^k *as input, and outputs a CRS* crs.

$\mathsf{HBG}^{\mathsf{sdp}}.\mathsf{GenBits}(\mathsf{crs}) \xrightarrow{\$} (r, \mathsf{st})$: *The bits generation algorithm takes a CRS* crs *as input, and outputs a string* $r \in \{0,1\}^k$ *and a state* st.

$\mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Prove}(\mathsf{st}, I) \xrightarrow{\$} \pi$: *The proving algorithm takes a state* st *and a subset* $I \subseteq [k]$ *as input, and outputs a proof* π.

$\mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Verify}(\mathsf{crs}, I, r_I, \pi) \to \top$ *or* $\bot$: *The verification algorithm takes a CRS* crs, *a subset* $I \subseteq [k]$, *a string* $r_I \in \{0,1\}^{|I|}$, *and a proof* π *as input, and outputs* $\top$ *indicating acceptance or* $\bot$ *otherwise.*

We require an SDP-HBG to satisfy the following properties:

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

$$\Pr\left[\mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Verify}(\mathsf{crs}, I, r_I, \pi) = \top : \begin{array}{l} \mathsf{crs} \xleftarrow{\$} \mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Setup}(1^\lambda, 1^k); \\ (r, \mathsf{st}) \xleftarrow{\$} \mathsf{HBG}^{\mathsf{sdp}}.\mathsf{GenBits}(\mathsf{crs}); \\ \pi \xleftarrow{\$} \mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Prove}(\mathsf{st}, I) \end{array}\right] = 1.$$

Somewhat Computational Binding: *There exists a constant* $\gamma < 1$ *such that (1) for any polynomial* $k = k(\lambda)$ *and CRS* crs *generated by* $\mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Setup}(1^\lambda, 1^k)$, *there exists a subset* $\mathcal{V}^{\mathsf{crs}} \subseteq \{0,1\}^k$ *such that* $|\mathcal{V}^{\mathsf{crs}}| \le 2^{k^\gamma \mathsf{poly}(\lambda)}$, *and (2) for any PPT adversary* $\mathcal{A}$, *we have*

$$\Pr\left[r_I \notin \mathcal{V}_I^{\mathsf{crs}} \wedge \mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Verify}(\mathsf{crs}, I, r_I, \pi) = \top : \begin{array}{l} \mathsf{crs} \xleftarrow{\$} \mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Setup}(1^\lambda, 1^k); \\ (I, r_I, \pi) \xleftarrow{\$} \mathcal{A}(\mathsf{crs}) \end{array}\right] = \mathsf{negl}(\lambda),$$

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

Computational Hiding: *For any polynomial* $k = k(\lambda)$, $I \subseteq [k]$, *and PPT adversary* $\mathcal{A}$, *we have*

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

where $\mathsf{crs} \xleftarrow{\$} \mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Setup}(1^\lambda, 1^k)$, $(r, \mathsf{st}) \xleftarrow{\$} \mathsf{HBG}^{\mathsf{sdp}}.\mathsf{GenBits}(\mathsf{crs})$, $\pi \xleftarrow{\$} \mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Prove}(\mathsf{st}, I)$, *and* $r' \xleftarrow{\$} \{0,1\}^k$.

An SDP-HBG can be seen as a weaker variant of an ordinary HBG, in the sense that the former can be naturally constructed from the latter.

Lemma 5. *If there exists an HBG, then there exists an SDP-HBG.*

Proof. Given an ordinary HBG $(\mathsf{HBG.Setup}, \mathsf{HBG.GenBits}, \mathsf{HBG.Verify})$, we can construct an SDP-HBG $(\mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Setup}, \mathsf{HBG}^{\mathsf{sdp}}.\mathsf{GenBits}, \mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Prove}, \mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Verify})$ as described in Fig. 2.

For the above construction of an SDP-HBG, it is straightforward to see that correctness and computational hiding follow from those of the underlying HBG, respectively. In particular, for computational hiding, the input of an adversary for the above SDP-HBG and an adversary for the underlying HBG take the same set of information, and thus constructing a reduction is trivial.

It remains to see the somewhat computational binding of the above SDP-HBG. By the succinct commitment property of the underlying HBG, there exist a constant $\gamma < 1$ and a polynomial $p = p(\lambda) = \mathsf{poly}(\lambda)$ such that $|\mathsf{com}| \le k^\gamma \cdot p$ holds for all com generated by $\mathsf{HBG.GenBits}(\mathsf{crs})$ where crs is in turn generated from $\mathsf{HBG.Setup}(1^\lambda, 1^k)$.

$\mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Setup}(1^\lambda, 1^k)$: $\mathsf{crs} \xleftarrow{\$} \mathsf{HBG.Setup}(1^\lambda, 1^k)$ Return crs.	$\mathsf{HBG}^{\mathsf{sdp}}.\mathsf{GenBits}(\mathsf{crs})$: $(\mathsf{com}, r, \{\pi_i\}_{i\in[k]}) \xleftarrow{\$} \mathsf{HBG.GenBits}(\mathsf{crs})$ Return $(r, \mathsf{st} = (\mathsf{com}, \{\pi_i\}_{i\in[k]}))$.
$\mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Prove}(\mathsf{st}, I)$: $(\mathsf{com}, \{\pi_i\}_{i\in[k]}) \leftarrow \mathsf{st}$ Return $\pi' = (\mathsf{com}, \{\pi_i\}_{i\in I})$.	$\mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Verify}(\mathsf{crs}, I, r_I, \pi')$: $(\mathsf{com}, \{\pi_i\}_{i\in I}) \leftarrow \pi'$ Return $\begin{cases} \top & \text{if } \forall i \in I : \mathsf{HBG.Verify}(\mathsf{crs}, \mathsf{com}, i, r_i, \pi_i) = \top \\ \bot & \text{otherwise} \end{cases}$.

Figure 2: The construction of an SDP-HBG from an HBG.

We will show that this constant γ satisfies the two required properties of the somewhat computational binding. For the first property, for a fixed crs generated from $\mathsf{HBG.Setup}(1^\lambda, 1^k)$, we define the set $\mathcal{V}^{\mathsf{crs}}$ by

$$\mathcal{V}^{\mathsf{crs}} := \left\{ r \in \{0,1\}^k | r = \mathsf{Open}(1^k, \mathsf{crs}, \mathsf{com}) \wedge \mathsf{com} \in \{0,1\}^{k^\gamma \cdot p} \right\},$$

where Open is the inefficient deterministic algorithm regarding the binding property of the underlying HBG. Then, $|\mathcal{V}^{\mathsf{crs}}| \leq 2^{k^\gamma \cdot p} = 2^{k^\gamma \cdot \mathsf{poly}(\lambda)}$ holds by the definition of γ.

To see that the second property is satisfied, consider a PPT adversary $\mathcal{A}$ against the somewhat computational binding of the SDP-HBG that on input a CRS crs (output from $\mathsf{HBG.Setup}(1^\lambda, 1^k)$) outputs $(I, r_I, \pi' = (\mathsf{com}, \{\pi_i\}_{i\in I}))$ satisfying the success condition, namely, $r_I \notin \mathcal{V}^{\mathsf{crs}}$ and $\mathsf{HBG.Verify}(\mathsf{crs}, \mathsf{com}, i, r_i, \pi_i) = \top$ for all $i \in I$. Let $r' := \mathsf{Open}(1^k, \mathsf{crs}, \mathsf{com})$. Then, note that the condition "$r_I \notin \mathcal{V}_I^{\mathsf{crs}}$" implies that there exists $i \in I$ for which $r_i \neq r'_i$ holds. Although the index i is not guaranteed to be efficiently computable, it can be simply guessed, which is correct with probability at least $1/|I| \geq 1/k$. Hence, we can straightforwardly construct a reduction algorithm $\mathcal{B}$ (attacking the binding property of the underlying HBG) that on input crs runs $(I, r_I, \pi' = (\mathsf{com}\{\pi_i\}_{i\in I})) \xleftarrow{\$} \mathcal{A}(\mathsf{crs})$, picks $i \in I$ uniformly at random, and outputs $(\mathsf{com}, i, r_i, \pi_i)$. Then, $\mathcal{B}$ is PPT, and its advantage in breaking the binding property of the underlying HBG is at least $1/k$ times $\mathcal{A}$'s advantage in breaking the somewhat computational binding property of the SDP-HBG. □

4.2 Construction

Here, we give a construction of an SDP-HBG from a SNARG and a OWF.

Theorem 3. *If there exist a OWF and a δ-slightly succinct SNARG for all* NP *languages for some $\delta < 1/2$ that satisfies adaptive computational soundness, then there exists an SDP-HBG that satisfies somewhat computational binding and computational hiding.*

Our construction of an SDP-HBG uses the following ingredients.

- An LR-wPRF $\mathcal{F} = \{F_K : \{0,1\}^m \to \{0,1\}\}_{K\in\{0,1\}^\kappa}$, built from a OWF via Theorem 1, with the key length $\kappa = \kappa(\lambda, \ell) = \ell \cdot \mathsf{poly}(\lambda)$, input length $m = m(\lambda, \ell) = \ell \cdot \mathsf{poly}(\lambda)$, and output length 1.
- A δ-slightly succinct SNARG $(\mathsf{SNARG.Setup}, \mathsf{SNARG.Prove}, \mathsf{SNARG.Verify})$ for some $\delta < 1/2$ for the language $\mathcal{L}$ associated with the relation $\mathcal{R}$ defined as follows:

$$\left((k', \{x_i\}_{i\in[k']}, \{r_i\}_{i\in[k']}),\ K\right) \in \mathcal{R} \iff r_i = F_K(x_i) \text{ for all } i \in [k'].$$

$\mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Setup}(1^\lambda, 1^k)$: $\mathsf{crs}_{\mathsf{snarg}} \xleftarrow{\$} \mathsf{SNARG.Setup}(1^\lambda)$ $\forall i \in [k] : x_i \xleftarrow{\$} \{0,1\}^m$ Return $\mathsf{crs} = (\mathsf{crs}_{\mathsf{snarg}}, \{x_i\}_{i\in[k]})$.	$\mathsf{HBG}^{\mathsf{sdp}}.\mathsf{GenBits}(\mathsf{crs})$: $(\mathsf{crs}_{\mathsf{snarg}}, \{x_i\}_{i\in[k]}) \leftarrow \mathsf{crs}$ $K \xleftarrow{\$} \{0,1\}^\kappa$ $\forall i \in [k] : r_i \leftarrow F_K(x_i)$ Return $(r = \{r_i\}_{i\in[k]}, \mathsf{st} = (\mathsf{crs}, K, r))$.
$\mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Prove}(\mathsf{st}, I)$: $(\mathsf{crs}, K, r) \leftarrow \mathsf{st}$ $(\mathsf{crs}_{\mathsf{snarg}}, \{x_i\}_{i\in[k]}) \leftarrow \mathsf{crs}$ $\pi \xleftarrow{\$} \mathsf{SNARG.Prove}(\mathsf{crs}_{\mathsf{snarg}}, (\lvert I\rvert, x_I, r_I), K)$ Return π.	$\mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Verify}(\mathsf{crs}, I, r_I, \pi)$: $(\mathsf{crs}_{\mathsf{snarg}}, \{x_i\}_{i\in[k]}) \leftarrow \mathsf{crs}$ Return $\mathsf{SNARG.Verify}(\mathsf{crs}_{\mathsf{snarg}}, (\lvert I\rvert, x_I, r_I), \pi)$.

Figure 3: The construction of an SDP-HBG based on an LR-wPRF and a SNARG.

In our construction of an SDP-HBG, the leakage bound ℓ of the underlying LR-wPRF is chosen depending on the length parameter k input to the setup algorithm of the SDP-HBG, so that

(a) $\kappa \le k^\gamma \cdot \mathsf{poly}(\lambda)$ holds for some constant $\gamma < 1$, and

(b) for any $k' \le k$, $x_i \in \{0,1\}^m$ for $i \in [k']$, $r_i \in \{0,1\}$ for $i \in [k']$, and $K \in \{0,1\}^\kappa$, if we generate $\mathsf{crs}_{\mathsf{snarg}} \xleftarrow{\$} \mathsf{SNARG.Setup}(1^\lambda)$ and $\pi \xleftarrow{\$} \mathsf{SNARG.Prove}(\mathsf{crs}_{\mathsf{snarg}}, (k', \{x_i\}_{i\in[k']}, \{r_i\}_{i\in[k']}), K)$, then we have $|\pi| \le \ell$.

Below we explain how we choose such ℓ.

Recall that the δ-slight succinctness of the SNARG ensures that the size of a proof π generated from a statement/witness pair $(x, w) \in \mathcal{R}$ satisfies $|\pi| \le (|x| + |w|)^\delta \cdot \mathsf{poly}(\lambda)$. In our case, the bit-length of a statement $x = (k', \{x_i\}_{i\in[k']}, \{r_i\}_{i\in[k']})$ is bounded by $\log k + k \cdot (m+1) \le k \cdot (m+2) = k\ell \cdot \mathsf{poly}(\lambda)$, and the bit-length of a witness $w = K$ is just $\kappa = \ell \cdot \mathsf{poly}(\lambda)$. Hence, the size of a proof π generated by $\mathsf{SNARG.Prove}$ for (x, w) is bounded by

$$|\pi| \le (k\ell \cdot \mathsf{poly}(\lambda) + \ell \cdot \mathsf{poly}(\lambda))^\delta \cdot \mathsf{poly}(\lambda) \le (k\ell)^\delta \cdot p,$$

for some polynomial $p = \mathsf{poly}(\lambda)$ that is independent of k and ℓ.

Then we set the leakage bound $\ell = \ell(\lambda, k)$ as

$$\ell := k^{\frac{\delta}{1-\delta}} \cdot p^{\frac{1}{1-\delta}}.$$

Since we assume $\delta < 1/2$, we have $\frac{\delta}{1-\delta} < 1$. Thus the property (a) is satisfied with $\gamma := \frac{\delta}{1-\delta}$. Furthermore, we have

$$|\pi| \le (k\ell)^\delta \cdot p = k^{\frac{\delta}{1-\delta}} p^{\frac{1}{1-\delta}} = \ell.$$

Hence, the property (b) is also satisfied, as desired.

Using the above ingredients, our construction of an SDP-HBG ($\mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Setup}, \mathsf{HBG}^{\mathsf{sdp}}.\mathsf{GenBits}, \mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Prove}, \mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Verify}$) is described in Fig. 3. In the description, x_I is a short hand for $\{x_i\}_{i\in I}$.

It is easy to see that the construction satisfies correctness. The security properties of the SDP-HBG are guaranteed by the following theorem.

Theorem 4. *The above SDP-HBG satisfies somewhat computational binding and computational hiding.*

Proof. We start by showing somewhat computational binding, then computational hiding.

Somewhat Computational Binding. For a CRS $\mathsf{crs} = (\mathsf{crs}_{\mathsf{snarg}}, \{x_i\}_{i\in[k]})$ output from $\mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Setup}(1^\lambda, 1^k)$, we define $\mathcal{V}^{\mathsf{crs}} := \{(F_K(x_1), ..., F_K(x_k)) \mid K \in \{0,1\}^\kappa\}$. Then, since $|K| = \kappa \leq k^\gamma \mathsf{poly}(\lambda)$, we have $|\mathcal{V}^{\mathsf{crs}}| \leq 2^{k^\gamma \mathsf{poly}(\lambda)}$. Furthermore, it is straightforward to see that by the soundness of the underlying SNARG, no PPT adversary can generate a valid proof for (I, r_I) that is inconsistent with any element of $\mathcal{V}$. More specifically, any PPT adversary that given $\mathsf{crs} = (\mathsf{crs}_{\mathsf{snarg}}, \{x_i\}_{i\in[k]})$ outputs a tuple (I, r_I, π) satisfying $\mathsf{SNARG.Verify}(\mathsf{crs}_{\mathsf{snarg}}, (|I|, x_I, r_I), \pi) = \top$ and $r_I \notin \mathcal{V}_I^{\mathsf{crs}}$, can be straightforwardly turned into a PPT adversary that breaks the adaptive soundness of the underlying SNARG, since $r_I \notin \mathcal{V}_I^{\mathsf{crs}}$ implies $(|I|, x_I, r_I) \notin \mathcal{L}$.

Computational Hiding. It is easy to reduce the computational hiding of our SDP-HBG to the security of the underlying LR-wPRF in which the leakage bound is ℓ, by noting that the leakage from K is π whose size is at most ℓ. Formally, given any polynomial $k = k(\lambda)$, $I \subseteq [k]$, and PPT adversary $\mathcal{A}$, consider the following PPT adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ that attacks the security of the underlying LR-wPRF with the leakage bound ℓ.

$\mathcal{B}_1^{F_K(\$),\mathsf{Leak}(\cdot)}(1^\lambda)$**:** (where $K \xleftarrow{\$} \{0,1\}^\kappa$) $\mathcal{B}_1$ makes $|I|$ queries to the oracle $F_K(\$)$, and regards the returned values from the oracle as $\{(x_i, r_i = F_K(x_i))\}_{i\in[I]}$. Next, $\mathcal{B}_1$ computes $\mathsf{crs}_{\mathsf{snarg}} \xleftarrow{\$} \mathsf{SNARG.Setup}(1^\lambda)$, and defines the circuit $f : \{0,1\}^\kappa \to \{0,1\}^\ell$ by $f(\cdot) := \mathsf{SNARG.Prove}(\mathsf{crs}_{\mathsf{crs}}, (|I|, x_I, r_I), \cdot)$. Then, $\mathcal{B}_1$ submits $f(\cdot)$ to the oracle $\mathsf{Leak}(\cdot)$, and receives π. Finally, $\mathcal{B}_1$ sets $\mathsf{st}_{\mathcal{B}}$ as all the information known to $\mathcal{B}_1$, and terminates with output $\mathsf{st}_{\mathcal{B}}$.

$\mathcal{B}_2^{\mathsf{Chal}(\$)}(\mathsf{st}_{\mathcal{B}})$**:** $\mathcal{B}_2$ submits $k-|I|$ queries to the challenge oracle $\mathsf{Chal}(\$)$, and regards the returned values from the oracle as $\{(x_i, r_i)\}_{i\in\overline{I}}$. Note that $r_i = F_K(x_i)$ if $b = 0$ and $r_i \xleftarrow{\$} \{0,1\}$ if $b = 1$, where b is $\mathcal{B}$'s challenge bit. Now, $\mathcal{B}_2$ sets $\mathsf{crs} := (\mathsf{crs}_{\mathsf{snarg}}, \{x_i\}_{i\in[k]})$, and runs $\mathcal{A}(\mathsf{crs}, I, r_I, \pi, r_{\overline{I}})$. When $\mathcal{A}$ terminates with output b', $\mathcal{B}_2$ outputs b' and terminates.

Since $|f(\cdot)| = \ell$, $\mathcal{B}$ complies with the rule of the LR-wPRF security experiment with the leakage bound ℓ. Furthermore, it is straightforward to see that if $b = 0$, then the pairs $\{(x_i, r_i)\}_{i\in\overline{I}}$ that $\mathcal{B}_2$ receives from the challenge oracle satisfy $F_K(x_i) = r_i$, and $\mathcal{B}$ simulates the computational hiding experiment in the case $r_{\overline{I}}$ is the real randomness generated by $\mathsf{HBG}^{\mathsf{sdp}}.\mathsf{GenBits}(\mathsf{crs})$, perfectly for $\mathcal{A}$. On the other hand, if $b = 1$, then $\{r_i\}_{i\in\overline{I}}$ are random bits, and $\mathcal{B}_2$ simulates the experiment of the opposite case (i.e. $r_{\overline{I}} = \{r_i\}_{i\in\overline{I}}$ is random) perfectly for $\mathcal{A}$. Hence, $\mathcal{B}$'s advantage in breaking the security of the underlying LR-wPRF is exactly the same as $\mathcal{A}$'s advantage in breaking the computational hiding property of our SDP-HBG. □

5 NIZK from SDP-HBG

In this section, we show the following theorem.

Theorem 5. *If there exists an SDP-HBG, then there exists a NIZK for all* NP *languages that satisfies adaptive computational soundness and non-adaptive single-theorem zero-knowledge.*

Combining Theorems 2, 3 and 5, we obtain the following theorem.

Theorem 6. *If there exist a OWF and a δ-slightly succinct SNARG for all* NP *languages for some $\delta < 1/2$ that satisfies adaptive computational soundness, then there exists a NIZK for all* NP *languages that satisfies adaptive soundness and adaptive multi-theorem zero-knowledge.*

In the following, we prove Theorem 5. The construction of our NIZK is almost the same as the scheme by Quach, Rothblum, and Wichs [QRW19], except that we use an SDP-HBG instead of an HBG.

$\mathsf{NIZK.Setup}(1^\lambda)$:	$\mathsf{NIZK.Prove}(\mathsf{crs}, x, w)$:
$\mathsf{crs}_{\mathsf{bgen}} \xleftarrow{\$} \mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Setup}(1^\lambda, 1^k)$	$(\mathsf{crs}_{\mathsf{bgen}}, s) \leftarrow \mathsf{crs}$
$s \xleftarrow{\$} \{0,1\}^k$	$(r, \mathsf{st}) \xleftarrow{\$} \mathsf{HBG}^{\mathsf{sdp}}.\mathsf{GenBits}(\mathsf{crs}_{\mathsf{bgen}})$
Return $\mathsf{crs} := (\mathsf{crs}_{\mathsf{bgen}}, s)$.	$\rho \leftarrow s \oplus r$
	$(I, \pi_{\mathsf{hbm}}) \xleftarrow{\$} \mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Prove}(\rho, x, w)$
	$\pi_{\mathsf{bgen}} \xleftarrow{\$} \mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Prove}(\mathsf{st}, I)$
	Return $\pi := (I, \pi_{\mathsf{hbm}}, r_I, \pi_{\mathsf{bgen}})$.
$\mathsf{NIZK.Verify}(\mathsf{crs}, x, \pi)$:	
$(\mathsf{crs}_{\mathsf{bgen}}, s) \leftarrow \mathsf{crs}$	
$(I, \pi_{\mathsf{hbm}}, r_I, \pi_{\mathsf{bgen}}) \leftarrow \pi$	
$\rho_I \leftarrow s_I \oplus r_I$	
If **(a)** $\wedge$ **(b)** then return $\top$ else return $\bot$:	
– **(a)** $\mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Verify}(I, \rho_I, x, \pi_{\mathsf{hbm}}) = \top$	
– **(b)** $\mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Verify}(\mathsf{crs}_{\mathsf{bgen}}, I, r_I, \pi_{\mathsf{bgen}}) = \top$	

Figure 4: The construction of a NIZK based on an SDP-HBG and an HBM-NIZK.

Construction. Our NIZK uses the following ingredients:

- An SDP-HBG $(\mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Setup}, \mathsf{HBG}^{\mathsf{sdp}}.\mathsf{GenBits}, \mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Prove}, \mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Verify})$.
- An HBM-NIZK $(\mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Prove}, \mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Verify})$ for an NP language $\mathcal{L}$ with ϵ-soundness.

Let $\gamma < 1$ be the constant regarding the somewhat computational binding of the SDP-HBG, which satisfies $|\mathcal{V}^{\mathsf{crs}_{\mathsf{bgen}}}| \leq 2^{k^\gamma \mathsf{poly}(\lambda)}$ for all $\mathsf{crs}_{\mathsf{bgen}}$ generated by $\mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Setup}(1^\lambda, 1^k)$. When we use an HBM-NIZK with the random-string length k, we can make $\epsilon = 2^{-\Omega(k)}$ as stated in Lemma 2. Therefore, we can take $k = k(\lambda) = \mathsf{poly}(\lambda)$ so that $|\mathcal{V}^{\mathsf{crs}}| \cdot \epsilon = \mathsf{negl}(\lambda)$ holds. We fix such k in the following.

Then, our construction of a NIZK for $\mathcal{L}$ is described in Figure 4.

It is easy to see that the construction satisfies completeness. The security properties of the NIZK is guaranteed by the following theorem.

Theorem 7. *The above NIZK satisfies adaptive computational soundness and non-adaptive single-theorem zero-knowledge.*

Proof. We start by showing soundness, and then zero-knowledge.

Adaptive Computational Soundness. Let $\mathcal{A}$ be any PPT adversary that attacks the adaptive soundness of our NIZK. Let Win be the event that $\mathcal{A}$ succeeds in breaking the adaptive soundness (i.e. $\mathsf{NIZK.Verify}(\mathsf{crs}, x, \pi) = \top$ and $x \notin \mathcal{L}$). Suppose $\mathcal{A}$ on input $\mathsf{crs} = (\mathsf{crs}_{\mathsf{bgen}}, s)$ outputs a pair $(x, \pi = (I, \pi_{\mathsf{hbm}}, r_I, \pi_{\mathsf{bgen}}))$. Let $\mathcal{V}^{\mathsf{crs}_{\mathsf{bgen}}} \subseteq \{0,1\}^k$ be the set with which the somewhat computational binding of the underlying SDP-HBG is considered. We have

$$\Pr[\mathsf{Win}] = \Pr[\mathsf{Win} \wedge r_I \notin \mathcal{V}_I^{\mathsf{crs}_{\mathsf{bgen}}}] + \Pr[\mathsf{Win} \wedge r_I \in \mathcal{V}_I^{\mathsf{crs}_{\mathsf{bgen}}}].$$

It is straightforward to see that $\Pr[\mathsf{Win} \wedge r_I \notin \mathcal{V}_I^{\mathsf{crs}_{\mathsf{bgen}}}] = \mathsf{negl}(\lambda)$ holds by the somewhat computational binding of the underlying SDP-HBG.

Hence, it remains to show $P := \Pr[\mathsf{Win} \wedge r_I \in \mathcal{V}_I^{\mathsf{crs}_{\mathsf{bgen}}}] = \mathsf{negl}(\lambda)$. Fix $\mathsf{crs}^*_{\mathsf{bgen}}$ in the image of $\mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Setup}(1^\lambda, 1^k)$ and $\mathcal{A}$'s randomness $r^*_{\mathcal{A}}$ that maximize the above probability P. Let $\mathcal{V}^* := \mathcal{V}^{\mathsf{crs}^*_{\mathsf{bgen}}}$. Let $F(\cdot)$ be the function that on input $s \in \{0,1\}^k$, computes $(x, \pi = (I, \pi_{\mathsf{hbm}}, r_I, \pi_{\mathsf{bgen}})) \leftarrow \mathcal{A}(\mathsf{crs} = (\mathsf{crs}^*_{\mathsf{bgen}}, s); r^*_{\mathcal{A}})$, and outputs $(x, I, \pi_{\mathsf{hbm}}, r_I)$. (Looking

$\mathcal{S}(1^\lambda, x)$:
$(I, \rho_I, \pi_{\mathsf{hbm}}) \xleftarrow{\$} \mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Sim}(x)$
$\mathsf{crs}_{\mathsf{bgen}} \xleftarrow{\$} \mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Setup}(1^\lambda, 1^k)$
$(r, \mathsf{st}) \xleftarrow{\$} \mathsf{HBG}^{\mathsf{sdp}}.\mathsf{GenBits}(\mathsf{crs}_{\mathsf{bgen}})$
$\pi_{\mathsf{bgen}} \xleftarrow{\$} \mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Prove}(\mathsf{st}, I)$
$s_I := \rho_I \oplus r_I$
$s_{\overline{I}} \xleftarrow{\$} \{0,1\}^{k-|I|}$
$\mathsf{crs} := (\mathsf{crs}_{\mathsf{bgen}}, s)$
$\pi := (I, \pi_{\mathsf{hbm}}, r_I, \pi_{\mathsf{bgen}})$
Return (crs, π).

Figure 5: The simulator for showing non-adaptive single-theorem zero-knowledge in the proof of Theorem 7.

ahead, F is essentially an adversary against the ϵ-soundness of the underlying HBM-NIZK.) Let P' be the following probability:

$$P' := \Pr\left[\begin{array}{l|l} \mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Verify}(I, s_I \oplus r_I, x, \pi_{\mathsf{hbm}}) = \top & s \xleftarrow{\$} \{0,1\}^k; \\ \wedge\ r_I \in \mathcal{V}_I^* \wedge x \notin \mathcal{L} & (x, I, \pi_{\mathsf{hbm}}, r_I) \leftarrow F(s) \end{array}\right].$$

Clearly, we have $P \le P'$. We also have

$$\begin{aligned} P' &= \sum_{r' \in \mathcal{V}^*} \Pr\left[\begin{array}{l|l} \mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Verify}(I, s_I \oplus r_I, x, \pi_{\mathsf{hbm}}) = \top & s \xleftarrow{\$} \{0,1\}^k; \\ \wedge\ r_I = r'_I \wedge x \notin \mathcal{L} & (x, I, \pi_{\mathsf{hbm}}, r_I) \leftarrow F(s) \end{array}\right] \\ &\le |\mathcal{V}^*| \cdot \max_{r^* \in \mathcal{V}^*} \Pr\left[\begin{array}{l|l} \mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Verify}(I, s_I \oplus r_I, x, \pi_{\mathsf{hbm}}) = \top & s \xleftarrow{\$} \{0,1\}^k; \\ \wedge\ r_I = r_I^* \wedge x \notin \mathcal{L} & (x, I, \pi_{\mathsf{hbm}}, r_I) \leftarrow F(s) \end{array}\right] \\ &= |\mathcal{V}^*| \cdot \max_{r^* \in \mathcal{V}^*} \Pr\left[\begin{array}{l|l} \mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Verify}(I, \rho_I, x, \pi_{\mathsf{hbm}}) = \top & \rho \xleftarrow{\$} \{0,1\}^k; \\ \wedge\ r_I = r_I^* \wedge x \notin \mathcal{L} & (x, I, \pi_{\mathsf{hbm}}, r_I) \leftarrow F(\rho \oplus r^*) \end{array}\right] \\ &\le |\mathcal{V}^*| \cdot \max_{r^* \in \mathcal{V}^*} \Pr\left[\begin{array}{l|l} \mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Verify}(I, \rho_I, x, \pi_{\mathsf{hbm}}) = \top & \rho \xleftarrow{\$} \{0,1\}^k; \\ \wedge\ x \notin \mathcal{L} & (x, I, \pi_{\mathsf{hbm}}, r_I) \leftarrow F(\rho \oplus r^*) \end{array}\right] \\ &\le |\mathcal{V}^*| \cdot \epsilon(k) = \mathsf{negl}(\lambda), \end{aligned}$$

where the last inequality uses the ϵ-soundness of the underlying HBM-NIZK which we consider for the adversary $\mathcal{B}(\rho)$ that outputs $F(\rho \oplus r^*)$ other than r_I, and the last equality is due to our choice of k. Hence, we have $P = \Pr[\mathsf{Win} \wedge r_I \in \mathcal{V}_I^{\mathsf{crs}_{\mathsf{bgen}}}] = \mathsf{negl}(\lambda)$ as well.

Combined together, we have seen that $\mathcal{A}$'s advantage in breaking the adaptive soundness of our NIZK is negligible. This implies that our NIZK satisfies adaptive soundness.

Non-Adaptive Single-Theorem Zero-Knowledge. Let $\mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Sim}$ be the simulator that is guaranteed to exist by the zero-knowledge of the underlying HBM-NIZK. Using $\mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Sim}$, we first give the description of the simulator $\mathcal{S}$ in Figure 5 for showing the non-adaptive single-theorem zero-knowledge of our NIZK.

We prove the non-adaptive single-theorem zero-knowledge of the above NIZK via a sequence of games argument using four games, among which the first game Game_1 (resp. the final game Game_4) is exactly the real (resp. simulated) experiment. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be any PPT adversary that attacks the non-adaptive single-theorem zero-knowledge of the above NIZK. For $j \in [4]$, let T_j be the event that $\mathcal{A}_2$ finally outputs 1 in Game_j. The description of the games is as follows.

Game_1: This is exactly the real experiment $\mathsf{Expt}^{\mathsf{nazk\text{-}real}}_{\mathcal{A}}$. We have $\Pr[\mathsf{T}_1] = \Pr[\mathsf{Expt}^{\mathsf{nazk\text{-}real}}_{\mathcal{A}}(\lambda) = 1]$.

Game_2: We change the ordering of the steps of Game_1, and furthermore "program" s by first choosing $\rho \xleftarrow{\$} \{0,1\}^k$ and then setting $s := \rho \oplus r$, without changing the distribution of $\mathcal{A}$'s view. Specifically, this game proceeds as follows.

1. Run $(x, w, \mathsf{st}_{\mathcal{A}}) \xleftarrow{\$} \mathcal{A}_1(1^\lambda)$.
2. Pick $\rho \xleftarrow{\$} \{0,1\}^k$.
3. Compute $(\pi_{\mathsf{hbm}}, I) \xleftarrow{\$} \mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Prove}(\rho, x, w)$.
4. Compute $\mathsf{crs}_{\mathsf{bgen}} \xleftarrow{\$} \mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Setup}(1^\lambda, 1^k)$.
5. Compute $(r, \mathsf{st}) \xleftarrow{\$} \mathsf{HBG}^{\mathsf{sdp}}.\mathsf{GenBits}(\mathsf{crs}_{\mathsf{bgen}})$.
6. Compute $\pi_{\mathsf{bgen}} \xleftarrow{\$} \mathsf{HBG}^{\mathsf{sdp}}.\mathsf{Prove}(\mathsf{st}, I)$.
7. Set $s_I := \rho_I \oplus r_I$.
8. Set $s_{\overline{I}} := \rho_{\overline{I}} \oplus r_{\overline{I}}$.[13]
9. Set $\mathsf{crs} := (\mathsf{crs}_{\mathsf{bgen}}, s)$ and $\pi := (I, \pi_{\mathsf{hbm}}, r_I, \pi_{\mathsf{bgen}})$.
10. Run $b' \xleftarrow{\$} \mathcal{A}_2(\mathsf{crs}, \pi, \mathsf{st}_{\mathcal{A}})$.

It is easy to see that the distribution of $\mathcal{A}$'s view has not been changed from Game_1. Hence, we have $\Pr[\mathsf{T}_1] = \Pr[\mathsf{T}_2]$.

Game_3: This game is identical to the previous game, except that the 8th step "$s_{\overline{I}} := \rho_{\overline{I}} \oplus r_{\overline{I}}$" is replaced with "$s_{\overline{I}} \xleftarrow{\$} \{0,1\}^{k-|I|}$".

It is straightforward to see that $|\Pr[\mathsf{T}_2] - \Pr[\mathsf{T}_3]| = \mathsf{negl}(\lambda)$ holds by the computational hiding of the underlying SDP-HBG.

Game_4: This game is identical to the previous game, except that $(\rho, \pi_{\mathsf{hbm}}, I)$ is generated as $(\rho_I, \pi_{\mathsf{hbm}}, I) \xleftarrow{\$} \mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Sim}(x)$, instead of picking $\rho \xleftarrow{\$} \{0,1\}^k$ and then executing $(\pi_{\mathsf{hbm}}, I) \xleftarrow{\$} \mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Prove}(\rho, x, w)$.

It is immediate to see that $|\Pr[\mathsf{T}_3] - \Pr[\mathsf{T}_4]| = \mathsf{negl}(\lambda)$ holds by the zero-knowledge of the underlying HBM-NIZK.

It is also straightforward to see that Game_4 is identical to the simulated experiment $\mathsf{Expt}^{\mathsf{nazk\text{-}sim}}_{\mathcal{A},\mathcal{S}}$. Hence, we have $\Pr[\mathsf{T}_4] = \Pr[\mathsf{Expt}^{\mathsf{nazk\text{-}sim}}_{\mathcal{A},\mathcal{S}}(\lambda) = 1]$.

Combined together, $\mathcal{A}$'s advantage against the non-adaptive single-theorem zero-knowledge can be estimated as follows:

$$\begin{aligned}\left|\Pr[\mathsf{Expt}^{\mathsf{nazk\text{-}real}}_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Expt}^{\mathsf{nazk\text{-}sim}}_{A,\mathcal{S}}(\lambda) = 1]\right| &= \left|\Pr[\mathsf{T}_1] - \Pr[\mathsf{T}_4]\right| \\ &\leq \sum_{j\in[3]} \left|\Pr[\mathsf{T}_j] - \Pr[\mathsf{T}_{j+1}]\right| = \mathsf{negl}(\lambda).\end{aligned}$$

This proves that our NIZK is non-adaptive single-theorem zero-knowledge. □

[13] Splitting the step "$s := \rho \oplus r$" into Steps 7 and 8 is to make it easier to describe the change in the next game and also see the correspondence with the procedure of the simulator $\mathcal{S}$.

Remark 8. *(On adaptive zero-knowledge.) One may think that we can prove that the above construction satisfies adaptive single-theorem zero-knowledge by relying on the special zero-knowledge property of the underlying HBM-NIZK, since a similar statement is proven for the construction of a NIZK based on an (ordinary) HBG in [QRW19]. However, we believe that this is not possible. Roughly speaking, the problem comes from the fact that the SDP-HBG enables us to generate a proof π_{bgen} corresponding to a subset I only after I is fixed. This is in contrast to an HBG where we can generate π_i that certifies the i-th hidden bit for each $i \in [k]$ before I is fixed. Specifically, the proof of adaptive zero-knowledge from an HBG in [QRW19] crucially relies on the fact that if $I \subseteq I^*$, then a set of proofs $\{\pi_i\}_{i\in I}$ can be derived from $\{\pi_i\}_{i\in I^*}$ in a trivial manner. On the other hand, we do not have a similar property in SDP-HBG since it generates a proof for a subset I in one-shot instead of generating a proof in a bit-by-bit manner. We note that if SDP-HBG satisfies an adaptive version of computational hiding where an adversary can choose a subset I depending on a CRS $\mathsf{crs}_{\mathsf{bgen}}$, then we can prove the adaptive zero-knowledge of the above scheme relying on special zero-knowledge of HBM-NIZK. However, such an adaptive version of computational hiding cannot be proven by a similar proof to the one in Section 4.2 due to the fact that a leakage function cannot depend on a challenge input in the security game of LR-wPRF. Therefore, instead of directly proving that the above scheme satisfies adaptive zero-knowledge, we rely on the generic conversion from non-adaptive to adaptive zero-knowledge as stated in Theorem 2.*

6 Zero-Knowledge SNARG

In this section, we consider a zero-knowledge SNARG (zkSNARG) which is a SNARG that also satisfies the zero-knowledge property.

Bitansky et al. [BCC+17] gave a construction of a zkSNARG in the designated verifier model based on a SNARG (with efficient verification) in the designated verifier model, a NIZK argument of knowledge, and a circuit-private FHE scheme. As noted in [BCC+17], if we consider a publicly verifiable SNARG (which is the default notion of a SNARG in this paper), then we need not rely on FHE. Moreover, a NIZK argument of knowledge can be constructed by combining any NIZK and CPA secure PKE. Thus, we obtain the following theorem:

Lemma 6. *Assume that there exist a fully succinct SNARG for all* NP *languages with adaptive computational soundness and efficient verification, a NIZK for all* NP *languages with adaptive computational soundness and adaptive multi-theorem zero-knowledge, and a CPA secure PKE scheme. Then, there exists a fully succinct SNARG for all* NP *languages with adaptive computational soundness, adaptive multi-theorem zero-knowledge, and efficient verification.*

Though this lemma follows from a straightforward extension of existing works, we give the construction in Appendix B for completeness.

Combining Lemma 6 and Theorem 6, we obtain the following theorem.

Theorem 8. *If there exist a CPA secure PKE scheme and a fully succinct SNARG for all* NP *languages with adaptive computational soundness and efficient verification, then there exists a fully succinct SNARG for all* NP *languages with adaptive computational soundness, adaptive multi-theorem zero-knowledge, and efficient verification.*

Remark 9. *We cannot prove a similar statement for a SNARG without efficient verification since efficient verification is essential in the construction of a zkSNARG in Appendix B.*

References

[AGV09] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 474–495. Springer, Heidelberg, March 2009. 3

[BBBF18] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable delay functions. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 757–788. Springer, Heidelberg, August 2018. 2

[BC12] Nir Bitansky and Alessandro Chiesa. Succinct arguments from multi-prover interactive proofs and their efficiency benefits. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 255–272. Springer, Heidelberg, August 2012. 8

[BCC+17] Nir Bitansky, Ran Canetti, Alessandro Chiesa, Shafi Goldwasser, Huijia Lin, Aviad Rubinstein, and Eran Tromer. The hunting of the SNARK. *Journal of Cryptology*, 30(4):989–1066, October 2017. 2, 3, 8, 25

[BCG+14] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE Computer Society Press, May 2014. 2

[BCI+13] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 315–333. Springer, Heidelberg, March 2013. 8

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *20th ACM STOC*, pages 103–112. ACM Press, May 1988. 1, 8

[BFS20] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent SNARKs from DARK compilers. In Vincent Rijmen and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, LNCS, pages 677–706. Springer, Heidelberg, May 2020. 8

[BISW17] Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu. Lattice-based SNARGs and their application to more efficient obfuscation. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 247–277. Springer, Heidelberg, April / May 2017. 2, 8

[BISW18] Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu. Quasi-optimal SNARGs via linear multi-prover interactive proofs. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 222–255. Springer, Heidelberg, April / May 2018. 2, 8

[BMW03] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 614–629. Springer, Heidelberg, May 2003. 1

[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 Moses Charikar and Edith Cohen, editors, *51st ACM STOC*, pages 1082–1090. ACM Press, June 2019. 8

[CCRR18] Ran Canetti, Yilei Chen, Leonid Reyzin, and Ron D. Rothblum. Fiat-Shamir and correlation intractability from strong KDM-secure encryption. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 91–122. Springer, Heidelberg, April / May 2018. 8

[CHK07] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. *Journal of Cryptology*, 20(3):265–294, July 2007. 1, 8

[CHM+20] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In Vincent Rijmen and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, LNCS, pages 738–768. Springer, Heidelberg, May 2020. 8

[CKU20] Geoffroy Couteau, Shuichi Katsumata, and Bogdan Ursu. Non-interactive zero-knowledge in pairing-free groups from weaker assumptions. In Vincent Rijmen and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, LNCS, pages 442–471. Springer, Heidelberg, May 2020. 2, 8

[CL18] Ran Canetti and Amit Lichtenberg. Certifying trapdoor permutations, revisited. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part I*, volume 11239 of *LNCS*, pages 476–506. Springer, Heidelberg, November 2018. 8

[CMS19] Alessandro Chiesa, Peter Manohar, and Nicholas Spooner. Succinct arguments in the quantum random oracle model. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 1–29. Springer, Heidelberg, December 2019. 8

[COS20] Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. Fractal: Post-quantum and transparent recursive proofs from holography. In Vincent Rijmen and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, LNCS, pages 769–793. Springer, Heidelberg, May 2020. 8

[Cv91] David Chaum and Eugène van Heyst. Group signatures. In Donald W. Davies, editor, *EUROCRYPT'91*, volume 547 of *LNCS*, pages 257–265. Springer, Heidelberg, April 1991. 1

[DDN00] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM J. Comput.*, 30(2):391–437, 2000. 1

[DFMV13] Ivan Damgård, Sebastian Faust, Pratyay Mukherjee, and Daniele Venturi. Bounded tamper resilience: How to go beyond the algebraic barrier. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 140–160. Springer, Heidelberg, December 2013. 1

[DHLW10] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Efficient public-key cryptography in the presence of key leakage. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 613–631. Springer, Heidelberg, December 2010. 1

[DN07] Cynthia Dwork and Moni Naor. Zaps and their applications. *SIAM J. Comput.*, 36(6):1513–1543, 2007. 2, 7, 17

[EFKP20] Naomi Ephraim, Cody Freitag, Ilan Komargodski, and Rafael Pass. SPARKs: Succinct parallelizable arguments of knowledge. In Vincent Rijmen and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, LNCS, pages 707–737. Springer, Heidelberg, May 2020. 8

[FLS99] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple noninteractive zero knowledge proofs under general assumptions. *SIAM J. Comput.*, 29(1):1–28, 1999. 1, 3, 8, 12, 13

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987. 8

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 626–645. Springer, Heidelberg, May 2013. 2, 8

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987. 1

[GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, December 1994. 1

[Gol11] Oded Goldreich. Basing non-interactive zero-knowledge on (enhanced) trapdoor permutations: The state of the art. In Oded Goldreich, editor, *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, volume 6650 of *LNCS*, pages 406–421. Springer, 2011. 8

[GOS12] Jens Groth, Rafail Ostrovsky, and Amit Sahai. New techniques for noninteractive zero-knowledge. *J. ACM*, 59(3):11:1–11:35, 2012. 1, 8

[GR13] Oded Goldreich and Ron D. Rothblum. Enhancements of trapdoor permutations. *Journal of Cryptology*, 26(3):484–512, July 2013. 8

[Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 321–340. Springer, Heidelberg, December 2010. 2, 8

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016. 2, 8

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108. ACM Press, June 2011. 2, 12, 13

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999. 9, 10

[HLWW16] Carmit Hazay, Adriana López-Alt, Hoeteck Wee, and Daniel Wichs. Leakage-resilient cryptography from minimal assumptions. *Journal of Cryptology*, 29(3):514–551, July 2016. 5, 7, 15

[KNYY19] Shuichi Katsumata, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Exploring constructions of compact NIZKs from various assumptions. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 639–669. Springer, Heidelberg, August 2019. 8

[KRR17] Yael Tauman Kalai, Guy N. Rothblum, and Ron D. Rothblum. From obfuscation to the security of Fiat-Shamir for proofs. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 224–251. Springer, Heidelberg, August 2017. 8

[KV09] Jonathan Katz and Vinod Vaikuntanathan. Signature schemes with bounded leakage resilience. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 703–720. Springer, Heidelberg, December 2009. 1

[KW18] Sam Kim and David J. Wu. Multi-theorem preprocessing NIZKs from lattices. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 733–765. Springer, Heidelberg, August 2018. 8

[LPWW20] Benoît Libert, Alain Passelègue, Hoeteck Wee, and David J. Wu. New constructions of statistical NIZKs: Dual-mode DV-NIZKs and more. In Vincent Rijmen and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, LNCS, pages 410–441. Springer, Heidelberg, May 2020. 15

[Mic00] Silvio Micali. Computationally sound proofs. *SIAM J. Comput.*, 30(4):1253–1298, 2000. 1, 8

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, January 1991. 10

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM STOC*, pages 427–437. ACM Press, May 1990. 1

[Ps05] Rafael Pass and Abhi shelat. Unconditional characterizations of non-interactive zero-knowledge. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 118–134. Springer, Heidelberg, August 2005. 8, 12

[PS19] Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for NP from (plain) learning with errors. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 89–114. Springer, Heidelberg, August 2019. 1, 8

[QRW19] Willy Quach, Ron D. Rothblum, and Daniel Wichs. Reusable designated-verifier NIZKs for all NP from CDH. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part II*, volume 11477 of *LNCS*, pages 593–621. Springer, Heidelberg, May 2019. 3, 4, 7, 13, 14, 15, 16, 17, 21, 25, 33, 34

[RST01] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 552–565. Springer, Heidelberg, December 2001. 1

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th ACM STOC*, pages 475–484. ACM Press, May / June 2014. 8

[Val08] Paul Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 1–18. Springer, Heidelberg, March 2008.

[Wee05] Hoeteck Wee. On round-efficient argument systems. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP 2005*, volume 3580 of *LNCS*, pages 140–152. Springer, Heidelberg, July 2005. 8

A Omitted Contents in Section 3

A.1 HBG from Non-Adaptive NIZK

In this section, we prove Lemma 3. By Lemma 1, we can generically upgrade non-adaptive single-theorem zero-knowledge to non-adaptive multi-theorem zero-knowledge. Therefore, we can assume that we have a NIZK with non-adaptive multi-theorem zero-knowledge. Here, we focus only on the computational case (adaptive computational soundness for the underlying NIZK and computational binding for the resulting HBG). The statistical case can be proved similarly.

We construct an HBG based on the following ingredients:

- A PRG $\mathsf{PRG} : \{0,1\}^\lambda \to \{0,1\}^k$.[14]
- A statistically binding non-interactive commitment scheme $(\mathsf{Com.Setup}, \mathsf{Com.Commit})$ in the CRS model with the message space $\{0,1\}^\lambda$. We denote the randomness space of $\mathsf{Com.Commit}$ by $\mathcal{R}_{\mathsf{com}}$.
- A NIZK $(\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Verify})$, satisfying adaptive computational soundness and non-adaptive multi-theorem zero-knowledge, for a language $\mathcal{L}$ associated with the relation $\mathcal{R}$ defined below:

$$\Big((\mathsf{crs}_{\mathsf{com}}, \mathsf{com}, i, r_i), (s, r_{\mathsf{com}})\Big) \in \mathcal{R} \iff \begin{array}{l} \mathsf{com} = \mathsf{Com.Commit}(\mathsf{crs}_{\mathsf{com}}, s; r_{\mathsf{com}}) \\ \wedge\ r_i = \mathsf{PRG}_i(s) \end{array},$$

  where $\mathsf{PRG}_i(s)$ denotes the i-th bit of $\mathsf{PRG}(s)$.

Our construction of an HBG $(\mathsf{HBG.Setup}, \mathsf{HBG.GenBits}, \mathsf{HBG.Verify})$ is described in Fig. 6. The correctness of the scheme is easy to see.

Theorem 9. *The above HBG satisfies succinct commitment, computational binding, and computational hiding.*

Proof. We show each of the properties based on the properties of the underlying primitives.

Succinct Commitment. This immediately follows from the fact that the commitment size is $\mathsf{poly}(\lambda)$, which is independent of k.

Computational Binding. We define an (inefficient) algorithm Open as follows:

[14] The output length k should be set to be the length parameter given as an input of $\mathsf{HBG.Setup}$.

$\mathsf{HBG.Setup}(1^\lambda, 1^k)$:	$\mathsf{HBG.GenBits}(\mathsf{crs})$:
$\mathsf{crs}_{\mathsf{com}} \xleftarrow{\$} \mathsf{Com.Setup}(1^\lambda)$ $\mathsf{crs}_{\mathsf{nizk}} \xleftarrow{\$} \mathsf{NIZK.Setup}(1^\lambda)$ Return $\mathsf{crs} := (\mathsf{crs}_{\mathsf{com}}, \mathsf{crs}_{\mathsf{nizk}})$.	$(\mathsf{crs}_{\mathsf{com}}, \mathsf{crs}_{\mathsf{nizk}}) \leftarrow \mathsf{crs}$ $s \xleftarrow{\$} \{0,1\}^\lambda$ $r_{\mathsf{com}} \xleftarrow{\$} \mathcal{R}_{\mathsf{com}}$ $\mathsf{com} \leftarrow \mathsf{Com.Commit}(\mathsf{crs}_{\mathsf{com}}, s; r_{\mathsf{com}})$ $r \leftarrow \mathsf{PRG}(s)$. $\forall i \in [k] : \pi_i \xleftarrow{\$} \mathsf{NIZK.Prove}(\mathsf{crs}_{\mathsf{nizk}}, (\mathsf{crs}_{\mathsf{com}}, \mathsf{com}, i, r_i), (s, r_{\mathsf{com}}))$ Return $(\mathsf{com}, r, \{\pi_i\}_{i\in[k]})$.
$\mathsf{HBG.Verify}(\mathsf{crs}, \mathsf{com}, i, r_i, \pi_i)$: $(\mathsf{crs}_{\mathsf{com}}, \mathsf{crs}_{\mathsf{nizk}}) \leftarrow \mathsf{crs}$ Return $\mathsf{NIZK.Verify}(\mathsf{crs}_{\mathsf{nizk}}, (\mathsf{crs}_{\mathsf{com}}, \mathsf{com}, i, r_i), \pi_i)$.	

Figure 6: The construction of an HBG from a PRG, a statistically commitment scheme, and a NIZK.

$\mathsf{Open}(1^k, \mathsf{crs}, \mathsf{com})$**:** This algorithm parses $(\mathsf{crs}_{\mathsf{com}}, \mathsf{crs}_{\mathsf{nizk}}) \leftarrow \mathsf{crs}$ and finds $s \in \{0,1\}^k$ such that $\mathsf{com} = \mathsf{Com.Commit}(\mathsf{crs}_{\mathsf{com}}, s; r_{\mathsf{com}})$ holds for some $r_{\mathsf{com}} \in \mathcal{R}_{\mathsf{com}}$ by brute-force. If there does not exist such s or exists multiple such s, it immediately aborts. Otherwise it outputs s.

Assume towards a contradiction that there exists a PPT adversary $\mathcal{A}$ that breaks the computational binding of the HBG w.r.t. the above defined Open with non-negligible probability. Then we construct a PPT adversary $\mathcal{B}$ that breaks the adaptive computational soundness of the underlying NIZK as follows:

$\mathcal{B}(\mathsf{crs}_{\mathsf{nizk}})$**:** This algorithm generates $\mathsf{crs}_{\mathsf{com}} \xleftarrow{\$} \mathsf{Com.Setup}(1^\lambda)$, sets $\mathsf{crs} := (\mathsf{crs}_{\mathsf{com}}, \mathsf{crs}_{\mathsf{nizk}})$, and runs $\mathcal{A}(\mathsf{crs})$ to obtain $(\mathsf{com}, i, r_i^*, \pi_i)$. Then it outputs a statement $(\mathsf{crs}_{\mathsf{com}}, \mathsf{com}, i, r_i^*)$ and a proof π_i.

This completes the description of $\mathcal{B}$. Since we are assuming that $\mathcal{A}$ breaks the computational binding of the HBG, we have that $\Pr[r_i^* \neq r_i \wedge \mathsf{HBG.Verify}(\mathsf{crs}, \mathsf{com}, i, r_i^*, \pi_i) = \top]$ is non-negligible where $r = \mathsf{Open}(1^k, \mathsf{crs}, \mathsf{com})$. Moreover, by the statistical binding of the underlying commitment scheme, for an overwhelming fraction of $\mathsf{crs}_{\mathsf{com}}$, any commitment com has at most one s such that there exists $r_{\mathsf{com}} \in \mathcal{R}_{\mathsf{com}}$ satisfying $\mathsf{com} = \mathsf{Com.Commit}(\mathsf{crs}_{\mathsf{com}}, s; r_{\mathsf{com}})$. We denote the event that $\mathcal{B}$ picks such $\mathsf{crs}_{\mathsf{com}}$ by Good. Then $\Pr[r_i^* \neq r_i \wedge \mathsf{HBG.Verify}(\mathsf{crs}, \mathsf{com}, i, r_i^*, \pi_i) = \top \wedge \mathsf{Good}]$ is non-negligible. In the following, we assume that the event

$$r_i^* \neq r_i \ \wedge \ \mathsf{HBG.Verify}(\mathsf{crs}, \mathsf{com}, i, r_i^*, \pi_i) = \top \ \wedge \ \mathsf{Good}$$

occurs. When this event occurs, it is clear that $(\mathsf{crs}_{\mathsf{com}}, \mathsf{com}, i, r_i^*) \notin \mathcal{L}$ holds. This means that $\mathcal{B}$ succeeds in breaking the adaptive computational soundness of the underlying NIZK, i.e., it succeeds in generating a valid proof π_i for a statement $(\mathsf{crs}_{\mathsf{com}}, \mathsf{com}, i, r_i^*) \notin \mathcal{L}$ with non-negligible probability. This contradicts the adaptive computational soundness of the underlying NIZK, and thus there exists no PPT adversary that can break the computational binding of the HBG.

Computational Hiding. Let $\mathcal{S}$ be the simulator for the non-adaptive multi-theorem zero-knowledge of the underlying NIZK. We fix k and $I \subset [k]$. We consider the following sequence of games for a PPT adversary $\mathcal{A}$:

Game_1**:** This is the original real game. That is, the game is described as follows.

1. Generate $\mathsf{crs}_{\mathsf{com}} \xleftarrow{\$} \mathsf{Com.Setup}(1^\lambda)$ and $\mathsf{crs}_{\mathsf{nizk}} \xleftarrow{\$} \mathsf{NIZK.Setup}(1^\lambda)$.

2. Choose $s \xleftarrow{\$} \{0,1\}^\lambda$.
3. Compute $\mathsf{com} \leftarrow \mathsf{Com.Commit}(\mathsf{crs}_{\mathsf{com}}, s; r_{\mathsf{com}})$ where $r_{\mathsf{com}} \xleftarrow{\$} \mathcal{R}_{\mathsf{com}}$.
4. Compute $r \leftarrow \mathsf{PRG}(s)$.
5. Generate $\pi_i \xleftarrow{\$} \mathsf{NIZK.Prove}(\mathsf{crs}_{\mathsf{nizk}}, (\mathsf{crs}_{\mathsf{com}}, \mathsf{com}, i, r_i), (s, r_{\mathsf{com}}))$ for all $i \in I$.
6. Output $\mathcal{A}((\mathsf{crs}_{\mathsf{com}}, \mathsf{crs}_{\mathsf{nizk}}), \mathsf{com}, I, r_I, \pi_I, r_{\overline{I}})$.

Game_2**:** This game is identical to the previous game except that $\mathsf{crs}_{\mathsf{nizk}}$ and π_I are generated by the simulator.

1. Generate $\mathsf{crs}_{\mathsf{com}} \xleftarrow{\$} \mathsf{Com.Setup}(1^\lambda)$.
2. Choose $s \xleftarrow{\$} \{0,1\}^\lambda$.
3. Compute $\mathsf{com} \leftarrow \mathsf{Com.Commit}(\mathsf{crs}_{\mathsf{com}}, s; r_{\mathsf{com}})$ where $r_{\mathsf{com}} \xleftarrow{\$} \mathcal{R}_{\mathsf{com}}$.
4. Compute $r \leftarrow \mathsf{PRG}(s)$.
5. Generate $(\mathsf{crs}_{\mathsf{nizk}}, \pi_I) \xleftarrow{\$} \mathcal{S}(1^\lambda, \{(\mathsf{crs}_{\mathsf{com}}, \mathsf{com}, i, r_i)\}_{i \in I})$.
6. Output $\mathcal{A}((\mathsf{crs}_{\mathsf{com}}, \mathsf{crs}_{\mathsf{nizk}}), \mathsf{com}, I, r_I, \pi_I, r_{\overline{I}})$.

This game is indistinguishable from the previous one by the non-adaptive multi-theorem zero-knowledge of the underlying NIZK. We remark that the non-adaptive zero-knowledge is sufficient since the statements are chosen independently of $\mathsf{crs}_{\mathsf{nizk}}$.

Game_3**:** This game is identical to the previous game except that com is replaced with a commitment of 0^λ.

1. Generate $\mathsf{crs}_{\mathsf{com}} \xleftarrow{\$} \mathsf{Com.Setup}(1^\lambda)$.
2. Choose $s \xleftarrow{\$} \{0,1\}^\lambda$.
3. Compute $\mathsf{com} \xleftarrow{\$} \mathsf{Com.Commit}(\mathsf{crs}_{\mathsf{com}}, 0^\lambda)$.
4. Compute $r \leftarrow \mathsf{PRG}(s)$.
5. Generate $(\mathsf{crs}_{\mathsf{nizk}}, \pi_I) \xleftarrow{\$} \mathcal{S}(1^\lambda, \{(\mathsf{crs}_{\mathsf{com}}, \mathsf{com}, i, r_i)\}_{i \in I})$.
6. Output $\mathcal{A}((\mathsf{crs}_{\mathsf{com}}, \mathsf{crs}_{\mathsf{nizk}}), \mathsf{com}, I, r_I, \pi_I, r_{\overline{I}})$.

This game is indistinguishable from the previous one by the computational hiding of the underlying commitment scheme.

Game_4**:** This game is identical to the previous game except that r is chosen uniformly at random from $\{0,1\}^k$.

1. Generate $\mathsf{crs}_{\mathsf{com}} \xleftarrow{\$} \mathsf{Com.Setup}(1^\lambda)$.
2. Choose $s \xleftarrow{\$} \{0,1\}^\lambda$.
3. Compute $\mathsf{com} \xleftarrow{\$} \mathsf{Com.Commit}(\mathsf{crs}_{\mathsf{com}}, 0^\lambda)$.
4. Generate $r \xleftarrow{\$} \{0,1\}^k$.
5. Generate $(\mathsf{crs}_{\mathsf{nizk}}, \pi_I) \xleftarrow{\$} \mathcal{S}(1^\lambda, \{(\mathsf{crs}_{\mathsf{com}}, \mathsf{com}, i, r_i)\}_{i \in I})$.
6. Output $\mathcal{A}((\mathsf{crs}_{\mathsf{com}}, \mathsf{crs}_{\mathsf{nizk}}), \mathsf{com}, I, r_I, \pi_I, r_{\overline{I}})$

This game is indistinguishable from the previous one by the security of the underlying PRG, noting that the seed s is no longer used due to the change made in Game_3.

Game_5**:** This game is identical to the previous game except that $\mathcal{A}$ is given $r'_{\overline{I}}$ instead of $r_{\overline{I}}$ where r' is a random string independent of r.

1. Generate $\mathsf{crs}_{\mathsf{com}} \xleftarrow{\$} \mathsf{Com.Setup}(1^\lambda)$.
2. Choose $s \xleftarrow{\$} \{0,1\}^\lambda$.
3. Compute $\mathsf{com} \xleftarrow{\$} \mathsf{Com.Commit}(\mathsf{crs}_{\mathsf{com}}, 0^\lambda)$.
4. Generate $r \xleftarrow{\$} \{0,1\}^k$ and $r' \xleftarrow{\$} \{0,1\}^k$.
5. Generate $(\mathsf{crs}_{\mathsf{nizk}}, \pi_I) \xleftarrow{\$} \mathcal{S}(1^\lambda, \{(\mathsf{crs}_{\mathsf{com}}, \mathsf{com}, i, r_i)\}_{i\in I})$.
6. Output $\mathcal{A}((\mathsf{crs}_{\mathsf{com}}, \mathsf{crs}_{\mathsf{nizk}}), \mathsf{com}, I, r_I, \pi_I, r'_{\overline{I}})$

This game is indistinguishable from the previous one since both r and r' are just random strings.

Game_6**:** This game is the ideal game.

1. Generate $\mathsf{crs}_{\mathsf{com}} \xleftarrow{\$} \mathsf{Com.Setup}(1^\lambda)$ and $\mathsf{crs}_{\mathsf{nizk}} \xleftarrow{\$} \mathsf{NIZK.Setup}(1^\lambda)$.
2. Choose $s \xleftarrow{\$} \{0,1\}^\lambda$.
3. Compute $\mathsf{com} \leftarrow \mathsf{Com.Commit}(\mathsf{crs}_{\mathsf{com}}, s; r_{\mathsf{com}})$ where $r_{\mathsf{com}} \xleftarrow{\$} \mathcal{R}_{\mathsf{com}}$.
4. Compute $r \leftarrow \mathsf{PRG}(s)$ and $r' \xleftarrow{\$} \{0,1\}^k$.
5. Generate $\pi_i \xleftarrow{\$} \mathsf{NIZK.Prove}(\mathsf{crs}_{\mathsf{nizk}}, (\mathsf{crs}_{\mathsf{com}}, \mathsf{com}, i, r_i), (s, r_{\mathsf{com}}))$ for all $i \in I$.
6. Output $\mathcal{A}((\mathsf{crs}_{\mathsf{com}}, \mathsf{crs}_{\mathsf{nizk}}), \mathsf{com}, I, r_I, \pi_I, r_{\overline{I}})$

This game is indistinguishable from the previous one by applying similar game hops from Game_1 to Game_4 in the reversed order.

Now, we have proved that Game_1 and Game_6 are computationally indistinguishable, which is exactly the requirement of the computational hiding. □

A.2 Adaptive NIZK from HBG

In this section, we prove Lemma 4. By Lemma 1, we can generically upgrade adaptive single-theorem zero-knowledge to adaptive multi-theorem zero-knowledge. Therefore, we only have to construct a NIZK with adaptive single-theorem zero-knowledge from an HBG. Here, we focus only on the computational case (where we assume the computational binding for the underlying HBG and prove the adaptive computational soundness for the resulting NIZK), since the statistical case was already proven in [QRW19].

The construction shown here is exactly the same as that in [QRW19]. Namely, we construct a NIZK based on the following ingredients.

- An HBG $(\mathsf{HBG.Setup}, \mathsf{HBG.GenBits}, \mathsf{HBG.Verify})$.
- An HBM-NIZK $(\mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Prove}, \mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Verify})$ for an NP language $\mathcal{L}$ with ϵ-soundness and special zero-knowledge.

By the succinct commitment property of the HBG, the commitment length $|\mathsf{com}|$ is at most $k^\gamma \mathsf{poly}(\lambda)$ for some constant $\gamma < 1$. On the other hand, when we use an HBM-NIZK with the random-string length k, we can make $\epsilon = 2^{-\Omega(k)}$ as stated in Lemma 2. Therefore, we can take $k = \mathsf{poly}(\lambda)$ so that $2^{|\mathsf{com}|}\epsilon = \mathsf{negl}(\lambda)$ holds. We fix such k.

$\mathsf{NIZK.Setup}(1^\lambda)$:	$\mathsf{NIZK.Prove}(\mathsf{crs}, x, w)$:
$\mathsf{crs}_{\mathsf{bgen}} \xleftarrow{\$} \mathsf{HBG.Setup}(1^\lambda, 1^k)$ $s \xleftarrow{\$} \{0,1\}^k$ Return $\mathsf{crs} := (\mathsf{crs}_{\mathsf{bgen}}, s)$.	$(\mathsf{crs}_{\mathsf{bgen}}, s) \leftarrow \mathsf{crs}$ $(\mathsf{com}, r, \{\pi_i\}_{i\in[k]}) \xleftarrow{\$} \mathsf{HBG.GenBits}(\mathsf{crs}_{\mathsf{bgen}})$ $\rho \leftarrow s \oplus r$ $(I, \pi_{\mathsf{hbm}}) \xleftarrow{\$} \mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Prove}(\rho, x, w)$ Return $\pi := (I, \pi_{\mathsf{hbm}}, \mathsf{com}, r_I, \pi_I)$.
$\mathsf{NIZK.Verify}(\mathsf{crs}, x, \pi)$: $(\mathsf{crs}_{\mathsf{bgen}}, s) \leftarrow \mathsf{crs}$ $(I, \pi_{\mathsf{hbm}}, r_I, \pi_I) \leftarrow \pi$ $\rho_I \leftarrow s_I \oplus r_I$ If **(a)** $\wedge$ **(b)** then return $\top$ else return $\bot$: – **(a)** $\mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Verify}(I, \rho_I, x, \pi_{\mathsf{hbm}}) = \top$ – **(b)** $\forall i \in I : \mathsf{HBG.Verify}(\mathsf{crs}_{\mathsf{bgen}}, \mathsf{com}, i, r_i, \pi_i) = \top$	

Figure 7: The construction of a NIZK based on an HBG and an HBM-NIZK.

Then the construction of a NIZK for $\mathcal{L}$ with adaptive single-theorem zero-knowledge is described in Figure 7. It is easy to see that the scheme satisfies the completeness.

In the following, we show the adaptive computational soundness and adaptive single-theorem zero-knowledge of the constructed NIZK.

Adaptive Computational Soundness. Let $\mathcal{A}$ be an adversary against the adaptive computational soundness of the constructed NIZK. We denote by Win the event that $\mathcal{A}$ wins, i.e., $\mathcal{A}$ is given crs and returns x^*, $\pi = (I, \pi_{\mathsf{hbm}}, r_I, \pi_I)$ such that $x^* \notin \mathcal{L}$ and $\mathsf{NIZK.Verify}(\mathsf{crs}, x^*, \pi) = \top$ hold. We denote by Bad the event that $r_I \neq \widetilde{r}_I$ holds where r_I is output by $\mathcal{A}$ and $\widetilde{r} := \mathsf{Open}(1^k, \mathsf{crs}_{\mathsf{bgen}}, \mathsf{com})$. Then, by the computational binding property of the underlying HBG, the probability that $\mathsf{HBG.Verify}(\mathsf{crs}_{\mathsf{bgen}}, \mathsf{com}, i, r_i, \pi_i) = \top$ holds for all $i \in [I]$ and Bad occurs, is negligible. In particular, we have $\Pr[\mathsf{Win} \wedge \mathsf{Bad}] = \mathsf{negl}(\lambda)$.

In the following, we prove $\Pr[\mathsf{Win} \wedge \overline{\mathsf{Bad}}] = \mathsf{negl}(\lambda)$. First, we show that $\Pr[\mathsf{Win} \wedge \overline{\mathsf{Bad}} \wedge \mathsf{com} = \widehat{\mathsf{com}}] \leq \epsilon$ holds for any fixed $\widehat{\mathsf{com}} \in \{0,1\}^{|\mathsf{com}|}$ and any $\mathsf{crs}_{\mathsf{bgen}}$ output by $\mathsf{HBG.Setup}(1^\lambda, 1^k)$. This can be seen by considering the following adversary $\mathcal{B}$ against the ϵ-soundness of the underlying HBM-NIZK:

$\mathcal{B}(\rho)$**:** This algorithm computes $\widetilde{r}_I := \mathsf{Open}(1^k, \mathsf{crs}_{\mathsf{bgen}}, \widehat{\mathsf{com}})$ and $s := \rho \oplus \widetilde{r}$, sets $\mathsf{crs} := (\mathsf{crs}_{\mathsf{bgen}}, s)$, runs $(x^*, \pi = (I, \pi_{\mathsf{hbm}}, \mathsf{com}, r_I, \pi_I)) \xleftarrow{\$} \mathcal{A}(\mathsf{crs})$, and returns $(x^*, \pi_{\mathsf{hbm}}, I)$.

In the above algorithm, if we assume that Bad does not occur and $\mathsf{com} = \widehat{\mathsf{com}}$, then we have $r_I = \widetilde{r}_I$. Therefore, we can see that $\mathcal{B}$ wins the game, i.e., $x^* \notin \mathcal{L}$ and $\mathsf{NIZK}^{\mathsf{hbm}}.\mathsf{Verify}(I, \rho_I, x, \pi_{\mathsf{hbm}}) = \top$, whenever the event $(\mathsf{Win} \wedge \overline{\mathsf{Bad}} \wedge \mathsf{com} = \widehat{\mathsf{com}})$ occurs. By the ϵ-soundness of the underlying HBM-NIZK, this event occurs with probability at most ϵ. By taking the union bound over all possible $\widehat{\mathsf{com}}$, we have $\Pr[\mathsf{Win} \wedge \overline{\mathsf{Bad}}] = \mathsf{negl}(\lambda) \leq 2^{|\mathsf{com}|}\epsilon = \mathsf{negl}(\lambda)$ due to our choice of k. In summary, $\Pr[\mathsf{Win}] = \mathsf{negl}(\lambda)$, which completes the proof of the adaptive computational soundness of the NIZK.

Adaptive Single-Theorem Zero-Knowledge. This can be reduced to the computational hiding of the underlying HBG and the special zero-knowledge of the underlying HBM-NIZK. We omit the detail since this is exactly the same as the proof in [QRW19].

$\mathsf{zkSNARG.Setup}(1^\lambda)$: $\mathsf{crs}_{\mathsf{snarg}} \xleftarrow{\$} \mathsf{SNARG.Setup}(1^\lambda)$ $\mathsf{crs}_{\mathsf{nizk}} \xleftarrow{\$} \mathsf{NIZK.Setup}(1^\lambda)$ $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{PKE.KeyGen}(1^\lambda)$ Return $\mathsf{crs}_{\mathsf{zksnarg}} := (\mathsf{crs}_{\mathsf{snarg}}, \mathsf{crs}_{\mathsf{nizk}}, \mathsf{pk})$.	$\mathsf{zkSNARG.Prove}(\mathsf{crs}_{\mathsf{zksnarg}}, x, w)$: $(\mathsf{crs}_{\mathsf{snarg}}, \mathsf{crs}_{\mathsf{nizk}}, \mathsf{pk}) \leftarrow \mathsf{crs}_{\mathsf{zksnarg}}$ $\pi_{\mathsf{snarg}} \xleftarrow{\$} \mathsf{SNARG.Prove}(\mathsf{crs}_{\mathsf{snarg}}, x, w)$ $r \xleftarrow{\$} \mathcal{R}$ $\mathsf{ct} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, \pi_{\mathsf{snarg}}; r)$ $\pi_{\mathsf{nizk}} \xleftarrow{\$} \mathsf{NIZK.Prove}(\mathsf{crs}_{\mathsf{nizk}}, (x, \mathsf{crs}_{\mathsf{snarg}}, \mathsf{pk}, \mathsf{ct}), (\pi_{\mathsf{snarg}}, r))$ Return $\pi_{\mathsf{zksnarg}} := (\mathsf{ct}, \pi_{\mathsf{nizk}})$.
$\mathsf{zkSNARG.Verify}(\mathsf{crs}_{\mathsf{zksnarg}}, x, \pi_{\mathsf{zksnarg}})$: $(\mathsf{crs}_{\mathsf{snarg}}, \mathsf{crs}_{\mathsf{nizk}}, \mathsf{pk}) \leftarrow \mathsf{crs}_{\mathsf{zksnarg}}$ $(\mathsf{ct}, \pi_{\mathsf{nizk}}) \leftarrow \pi_{\mathsf{zksnarg}}$ Return $\mathsf{NIZK.Verify}(\mathsf{crs}_{\mathsf{nizk}}, (x, \mathsf{crs}_{\mathsf{snarg}}, , \mathsf{pk}, \mathsf{ct}), \pi_{\mathsf{nizk}})$.	

Figure 8: The construction of a zkSNARG based on a SNARG, a NIZK, and a PKE scheme.

B Proof of Lemma 6

Here, we prove that we can construct a zkSNARG from a SNARG, a NIZK, and a PKE scheme. We construct a zkSNARG for an NP language $\mathcal{L}$ based on the following building blocks:

- A SNARG $(\mathsf{SNARG.Setup}, \mathsf{SNARG.Prove}, \mathsf{SNARG.Verify})$ for $\mathcal{L}$ with adaptive computational soundness and efficient verification.
- A CPA secure PKE scheme $(\mathsf{PKE.KeyGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$. We denote the randomness space of $\mathsf{PKE.Enc}$ by $\mathcal{R}$.
- A NIZK $(\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Verify})$ for the language $\mathcal{L}_{\mathsf{nizk}}$ associated with the relation $\mathcal{R}_{\mathsf{nizk}}$ defined below:

$$\Big((x, \mathsf{crs}_{\mathsf{snarg}}, \mathsf{pk}, \mathsf{ct}), (\pi_{\mathsf{snarg}}, r)\Big) \in \mathcal{R}_{\mathsf{nizk}} \iff \begin{array}{l} \mathsf{SNARG.Verify}(\mathsf{crs}_{\mathsf{snarg}}, x, \pi_{\mathsf{snarg}}) = \top \\ \wedge\ \mathsf{ct} = \mathsf{PKE.Enc}(\mathsf{pk}, \pi_{\mathsf{snarg}}; r) \end{array}.$$

  We require that the NIZK satisfy adaptive computational soundness and adaptive multi-theorem zero-knowledge.

Then, our construction of a zkSNARG $(\mathsf{zkSNARG.Setup}, \mathsf{zkSNARG.Prove}, \mathsf{zkSNARG.Verify})$ is given in Fig. 8.

Theorem 10. *The above non-interactive argument satisfies efficient verification, adaptive computational soundness, and adaptive multi-theorem zero-knowledge.*

Proof. **Efficient Verification.** By the full succinctness of the underlying SNARG, for all $(x, w) \in \mathcal{R}$, $\mathsf{crs}_{\mathsf{snarg}} \xleftarrow{\$} \mathsf{SNARG.Setup}(1^\lambda)$, and $\pi_{\mathsf{snarg}} \xleftarrow{\$} \mathsf{SNARG.Prove}(\mathsf{crs}_{\mathsf{snarg}}, x, w)$, the running time of $\mathsf{SNARG.Verify}(\mathsf{crs}_{\mathsf{snarg}}, x, \pi_{\mathsf{snarg}})$ is $\mathsf{poly}(\lambda)(|x| + |w|)^{o(1)}$. Especially, $|\pi_{\mathsf{snarg}}| = \mathsf{poly}(\lambda)(|x| + |w|)^{o(1)}$, and thus the running time of $\mathsf{PKE.Enc}(\mathsf{pk}, \pi_{\mathsf{snarg}}; r)$ is also $\mathsf{poly}(\lambda)(|x| + |w|)^{o(1)}$. Thus, the relation $\mathcal{R}_{\mathsf{nizk}}$ is verifiable in time $\mathsf{poly}(\lambda)(|x| + |w|)^{o(1)}$. Since the complexity of verification of the underlying NIZK is polynomial in the running time to verify the corresponding relation and the security parameter, $\mathsf{NIZK.Verify}(\mathsf{crs}_{\mathsf{nizk}}, (x, \mathsf{crs}_{\mathsf{snarg}}, , \mathsf{pk}, \mathsf{ct}), \pi_{\mathsf{nizk}})$ runs in time $\mathsf{poly}(\lambda)(|x| + |w|)^{o(1)}$.

Adaptive Computational Soundness. Let $\mathcal{A}$ be a PPT adversary against the adaptive computational soundness of the above construction. Then we construct a PPT adversary $\mathcal{B}$ against the adaptive computational soundness of the underlying SNARG as follows:

$\mathcal{B}(\mathsf{crs}_{\mathsf{snarg}})$**:** This algorithm generates $\mathsf{crs}_{\mathsf{nizk}} \xleftarrow{\$} \mathsf{NIZK.Setup}(1^\lambda)$ and $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{PKE.KeyGen}(1^\lambda)$, sets $\mathsf{crs}_{\mathsf{zksnarg}} := (\mathsf{crs}_{\mathsf{snarg}}, \mathsf{crs}_{\mathsf{nizk}}, \mathsf{pk})$, runs $(x^*, \pi_{\mathsf{zksnarg}} = (\mathsf{ct}, \pi_{\mathsf{nizk}})) \xleftarrow{\$} \mathcal{A}(\mathsf{crs}_{\mathsf{zksnarg}})$, computes $\pi_{\mathsf{snarg}} \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}, \mathsf{ct})$, and outputs $(x^*, \pi_{\mathsf{snarg}})$.

If $\mathcal{A}$ succeeds in breaking the soundness of our construction with non-negligible probability, then

$$\Pr[x^* \notin \mathcal{L} \wedge \mathsf{NIZK.Verify}(\mathsf{crs}_{\mathsf{nizk}}, (x^*, \mathsf{crs}_{\mathsf{snarg}}, \mathsf{pk}, \mathsf{ct}), \pi_{\mathsf{nizk}}) = \top]$$

is non-negligible. On the other hand, by the adaptive computational soundness of the underlying NIZK,

$$\Pr[\mathsf{NIZK.Verify}(\mathsf{crs}_{\mathsf{nizk}}, (x^*, \mathsf{crs}_{\mathsf{snarg}}, \mathsf{pk}, \mathsf{ct}), \pi_{\mathsf{nizk}}) = \top \wedge (x^*, \mathsf{crs}_{\mathsf{snarg}}, \mathsf{pk}, \mathsf{ct}) \notin \mathcal{L}_{\mathsf{nizk}}]$$

is negligible. Therefore,

$$\Pr[x^* \notin \mathcal{L} \wedge (x^*, \mathsf{crs}_{\mathsf{snarg}}, \mathsf{pk}, \mathsf{ct}) \notin \mathcal{L}_{\mathsf{nizk}}]$$

is non-negligible. Suppose that this event happens. Then by the definition of $\mathcal{L}_{\mathsf{nizk}}$, ct is an encryption of π_{snarg} that passes the verification of the underlying SNARG, in which case $\mathcal{B}$ clearly succeeds in breaking its adaptive soundness. Therefore, there exists no PPT adversary that breaks the adaptive computational soundness of our construction.

Adaptive Multi-theorem Zero-Knowledge. Let $\mathcal{S}_{\mathsf{nizk}} = (\mathcal{S}_{\mathsf{nizk},1}, \mathcal{S}_{\mathsf{nizk},2})$ be the simulator for the adaptive multi-theorem zero-knowledge of the underlying NIZK. Then we construct a simulator $\mathcal{S}_{\mathsf{zksnarg}} = (\mathcal{S}_{\mathsf{zksnarg},1}, \mathcal{S}_{\mathsf{zksnarg},2})$ for our construction as follows:

$\mathcal{S}_{\mathsf{zksnarg},1}(1^\lambda)$: This algorithm generates $(\mathsf{crs}_{\mathsf{nizk}}, \mathsf{st}_{\mathsf{nizk}}) \xleftarrow{\$} \mathcal{S}_{\mathsf{nizk},1}$, $\mathsf{crs}_{\mathsf{snarg}} \xleftarrow{\$} \mathsf{SNARG.Setup}(1^\lambda)$, and $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{PKE.KeyGen}(1^\lambda)$, and outputs $\mathsf{crs}_{\mathsf{zksnarg}} := (\mathsf{crs}_{\mathsf{snarg}}, \mathsf{crs}_{\mathsf{nizk}}, \mathsf{pk})$ and $\mathsf{st}_{\mathsf{zksnarg}} := (\mathsf{crs}_{\mathsf{zksnarg}}, \mathsf{st}_{\mathsf{nizk}})$.

$\mathcal{S}_{\mathsf{zksnarg},2}(\mathsf{st}_{\mathsf{zksnarg}}, x)$**:** This algorithm computes $\mathsf{ct} \xleftarrow{\$} \mathsf{PKE.Enc}(\mathsf{pk}, 0^{|\pi_{\mathsf{snarg}}|})$ and $\pi_{\mathsf{nizk}} \xleftarrow{\$} \mathcal{S}_{\mathsf{nizk},2}(\mathsf{st}_{\mathsf{nizk}}, (x, \mathsf{crs}_{\mathsf{snarg}}, , \mathsf{pk}, \mathsf{ct}))$, and outputs $\pi_{\mathsf{zksnarg}} := (\mathsf{ct}, \pi_{\mathsf{nizk}})$.

We can see that the CRS and proofs that are generated by the above simulator are indistinguishable from real ones by the zero-knowledge property of the underlying NIZK and the CPA security of the underlying PKE scheme. Since this is easy, we just give a sketch below. Starting from the real game, we consider a hybrid where $\mathsf{crs}_{\mathsf{nizk}}$ and π_{nizk} are generated by $\mathcal{S}_{\mathsf{nizk}}$ instead of being honestly generated. From the view point of any PPT adversary against adaptive multi-theorem zero-knowledge, this hybrid is indistinguishable from the real game by the adaptive multi-theorem zero-knowledge property of the underlying NIZK. Then, since the randomness r for an encryption of PKE is no longer used directly, we can replace $\mathsf{ct} \xleftarrow{\$} \mathsf{PKE.Enc}(\mathsf{pk}, \pi_{\mathsf{snarg}})$ with $\mathsf{ct} \xleftarrow{\$} \mathsf{PKE.Enc}(\mathsf{pk}, 0^{|\pi_{\mathsf{snarg}}|})$ by the CPA security of the underlying PKE scheme. At this point, the resulting hybrid is exactly the same as the simulated experiment using $\mathcal{S}_{\mathsf{zksnarg}}$. Thus, the real experiment and the simulated experiment are indistinguishable, which means that the construction satisfies adaptive multi-theorem zero-knowledge. □