Statistical ZAPs from Group-Based Assumptions

Geoffroy Couteau, Shuichi Katsumata, Elahe Sadeghi, Bogdan Ursu

Statistical ZAPs from Group-Based Assumptions

Geoffroy Couteau[1], Shuichi Katsumata[2], Elahe Sadeghi[3], and Bogdan Ursu[4]

[1] CNRS, IRIF, Université de Paris
geoffroy.couteau@ens.fr
[2] AIST, Japan
shuichi.katsumata@aist.go.jp
[3] Sharif University of Technology, Iran
e.sadeghi95@student.sharif.edu
[4] Department of Computer Science, ETH Zurich, Switzerland
bogdan.ursu@inf.ethz.ch

Abstract. We put forth a template for constructing statistical ZAPs for NP. Our template compiles NIZKs for NP in the hidden-bit model (which exist unconditionally) into statistical ZAPs using a new notion of *interactive hidden-bit generator* (IHBG), which adapts the notion of hidden-bit generator to the plain model by building upon the recent notion of statistically-hiding extractable commitments. We provide a construction of IHBG from the explicit hardness of the decision Diffie-Hellman assumption (where *explicit* refers to requiring an explicit upper bound on the advantage of any polynomial-time adversary against the assumption) and the existence of statistical ZAPs for a specific simple language, building upon the recent construction of dual-mode hidden-bit generator from (Libert et al., EUROCRYPT 2020). We provide two instantiations of the underlying simple ZAP:

- Using the recent statistical ZAP for the Diffie-Hellman language of (Couteau and Hartmann, CRYPTO 2020), we obtain statistical ZAPs for NP assuming (the explicit hardness of) DDH in $\mathbb{G}_1$ and kernel-DH in $\mathbb{G}_2$ (a *search* assumption which is weaker than DDH), where $(\mathbb{G}_1, \mathbb{G}_2)$ are groups equipped with an asymmetric pairing. This improves over the recent work of (Lombardi et al., EUROCRYPT 2020) which achieved a relaxed variant of statistical ZAP for NP, under a stronger assumption.
- Using the recent work of (Couteau et al., EUROCRYPT 2020), we obtain statistical ZAPs for NP assuming the explicit hardness of DDH, together with the assumption that no efficient adversary can break the key-dependent message one-wayness of ElGamal with respect to *efficient* functions over groups of size 2^λ with probability better than $\mathsf{poly}(\lambda)/2^{(c+o(1))\cdot\lambda}$, denoted $2^{-c\lambda}$-OW-KDM, for a constant $c = 1/2$, in pairing-free groups. Note that the latter is a search discrete-log-style falsifiable assumption, incomparable to DDH (in particular, it is not known to imply public-key encryption).

1 Introduction

Zero-knowledge proof systems, introduced in [21], are a fundamental cryptographic primitive, allowing a prover to convince a verifier of the veracity of a statement, while not divulging anything beyond whether the statement is true. Zero-knowledge proofs have countless applications. However, they suffer from strong lower bounds on the number of rounds of interactions required in their execution: they require at least three rounds of interactions [20]. Therefore, the dream result of proofs that consists of a single message from the prover to the verifier (NIZKs [4]) can only be achieved when assuming a trusted setup. Due to the importance of round-efficient zero-knowledge proofs, a large effort has been devoted to the construction of such proofs; yet, this trusted setup is often undesirable.

Witness-indistinguishability (WI) [18] is a natural relaxation of zero-knowledge, and is one of the most widely used privacy notions in proof systems. It provides the following guarantee: if there exist two witnesses (w_0, w_1) for a statement $x \in \mathscr{L}$, the verifier should not be able to distinguish an honest prover using w_0 from an honest prover using w_1. Witness-indistinguishable proofs can replace zero-knowledge proofs in many of their applications. At the same time, their round complexity is not subject to *any* known lower bounds.

ZAPs. The work of Dwork and Naor [13] introduced (and constructed) ZAPs, which are two-message public-coin WI proof systems. These proof systems have several advantages: being public-coin, they

are publicly verifiable (the validity of the proof can be verified solely by looking at the transcript). Furthermore, the first flow, which is just a uniformly random string, is inherently reusable for an arbitrary (polynomial) number of proofs on possibly different statements. ZAPs have proven to be important cryptographic primitives. By now, we have constructions of ZAPs from many standard assumptions, including trapdoor permutations (which is implied by factoring) [13], the decision linear assumption (DLIN) in bilinear maps [23], the (quasi-polynomial hardness of the) learning with error assumption [1,22,36], and also from more complex notions, such as indistinguishability obfuscation [3].

Statistical ZAP arguments. ZAPs were initially defined to satisfy unbounded soundness, and computational WI [13]. Statistical ZAP arguments provide the converse properties: computational soundness, and witness-indistinguishability against unbounded attackers. Unlike their computational WI counterpart, statistical ZAP arguments enjoy a very appealing property, that of *everlasting security*. Namely, soundness is an *online* security notion: as long as the prover cannot break soundness *at the time where it produces the proof*, security is guaranteed, even if the assumption it is based upon is later broken. On the other hand, WI and zero-knowledge should hold not only during the proof generation, but must continuously keep on holding in the future: compromising the assumptions underlying the WI property of proofs generated in the past at any point in the future would be sufficient to break privacy. Hence, targeting statistical privacy avoids being forced to assume the nonexistence of unforeseen cryptanalytic advances in the future.

Intriguingly, statistical ZAPs have proven much harder to construct than their computationally WI counterparts. In fact, for almost two decades after their introduction and until very recently, no construction of statistical ZAP argument was known, under any assumption. The situation changed very recently, with the construction of statistical ZAP arguments under the quasi-polynomial hardness of LWE, in two concurrent and independent works [1, 22]. Still, these results leave open the question of whether statistical ZAPs can be based on any of the other cryptographic assumptions that computational ZAPs can be based on, such as factoring or pairing-based assumptions.

The very recent work of [37] comes very close to improving this state of affairs: they construct, from the quasi-polynomial hardness of the decision linear assumption in bilinear groups, *ZAPs with private randomness*. This primitive is essentially as versatile as a standard ZAP: while the verifier uses private coins, the proof remains publicly verifiable, and the first flow remains reusable. Yet, it still falls short of constructing true statistical ZAPs from pairing-based assumptions.

1.1 Our Result

In this work, we develop a new approach for constructing statistical ZAPs. At a high-level, our approach works by bootstrapping statistical ZAPs for simple languages to statistical ZAPs for NP, using a new primitive called *interactive hidden-bits generator* (IHBG), a plain-model variant of hidden-bits generators, which have been recently introduced in [9, 31, 34, 41] for constructing NIZKs for NP from different assumptions. We provide two instantiations of our framework (in groups with or without pairings in the publicly verifiable setting), and obtain:

- **Statistical ZAPs in pairing groups.** A statistical ZAP argument for NP, assuming the explicit hardness[5] of the DDH assumption in $\mathbb{G}_1$ and of the kernel Diffie-Hellman assumption in $\mathbb{G}_2$, where $(\mathbb{G}_1, \mathbb{G}_2)$ are groups equipped with an asymmetric pairing. The kernel Diffie-Hellman assumption is a standard *search* assumption in bilinear groups [33,38], which is implied by (and is qualitatively weaker than) the DDH assumption. This improves over [37], both in terms of assumption (we rely on a qualitatively weaker assumption, since [37] requires DDH both in $\mathbb{G}_1$ and $\mathbb{G}_2$) and of the primitive constructed (we achieve a true statistical ZAP argument, while [37] achieves a relaxed variant).
- **Statistical ZAPs in pairing-free groups.** A statistical ZAP argument from NP, assuming explicit hardness of the DDH assumption in a *pairing-free* group $\mathbb{G}$ with $\log|\mathbb{G}| \approx \lambda^{1/2}$, and the assumption that no polynomial-time adversary can break the $\mathsf{OW\text{-}KDM}$ security of ElGamal with respect to efficient functions with success probability significantly better than $2^{-\lambda/2}$, denoted as

[5] *Explicit hardness* in [1] assumes that there exists an explicit bound μ on the advantage of any polynomial time adversary against the assumption. In particular, this is a weaker requirement than superpolynomial hardness, for any arbitrarily small superpolynomial function. We note that previous works on statistical ZAPs using quasi-polynomial hardness [1, 22, 36, 37] can instead use explicit hardness.

$2^{-\lambda/2}$ – OW-KDM security. Note that the best-known attack against such OW-KDM security of ElGamal succeeds with probability $\mathsf{poly}(\lambda) \cdot 2^{-\lambda}$. While non-standard, this is a *falsifiable* search assumption, and there is an exponential gap between the required security margin and the best known attack. Under the same KDM assumption, but assuming only the standard polynomial hardness of DDH, we also obtain statistical NIZKs (NISZKs) for NP in the common reference string (CRS) model (settling for computational NIZKs, we can further relax DDH to computational Diffie-Hellman). This builds upon and improves over the recent work of [10] which constructed *computational NIZK arguments* in the CRS model, under CDH and a stronger assumption: the $2^{-3\lambda/4}$-OW-KDM-hardness of ElGamal.

In all the above, the (decisional or kernel) Diffie-Hellman assumption can be replaced by any of its standard generalizations, namely the decisional k-Lin [27] and kernel k-Lin assumptions, or even more generally any assumption from the family of the (decisional or kernel) matrix Diffie-Hellman assumptions [15, 38].

Relation to [29]. In a breakthrough work (very recently accepted at Eurocrypt'21), Jain and Zhengzhong have solved the long-standing open problem of basing NIZKs on a well-studied assumption in pairing-free groups (the subexponential hardness of DDH). Furthermore, their work also achieves a statistical ZAP under the same assumption. We clarify the relation of our work to theirs.

The results presented in our work have been obtained concurrently and independently of those presented in [29]. However, we were made aware of the existence and content of [29] while it was submitted to Eurocrypt (through private communication), and before we had completed the write-up of our paper. The techniques developed in our work are unrelated to those in [29], and our results are complementary:

- We show that explicit hardness of DDH (or superpolynomial hardness of DDH, for any arbitrarily small superpolynomial function) gives statistical ZAPs in the pairing setting, and two-round statistical WI arguments in the pairing-free setting. In contrast, [29] relies on the subexponential hardness of DDH (but does not need pairings to achieve public verifiability).
- In the pairing-free setting, we also rely on an exponential search discrete-log-style hardness assumption, which is incomparable to subexponential DDH (albeit the latter is of course more standard). In particular, our assumption is falsifiable, holds in the generic group model, and is not known to imply public-key encryption.

Still, although our results have been achieved concurrently and independently of theirs, we cannot (and do not) claim to achieve the first construction of a statistical ZAP from standard group-based assumptions, since their construction precedes ours.

1.2 Our Techniques

At the heart of our results is a construction of a new cryptographic primitive, which we call an *interactive hidden-bits generator* (IHBG). At a high level, an IHBG adapts the notion of hidden-bits generator (defined in the CRS model) recently introduced and studied in [9, 31, 34, 41] to the plain model.

Dual-Mode Hidden-Bits Generators. More precisely, our starting point is the notion of a dual-mode hidden-bits generator (HBG) from [34]. In a dual-mode HBG, there are three algorithms: a CRS generation algorithm, a hidden-bits generator GenBits, and a verification algorithm VerifyBit. Given a CRS, the prover can, using GenBits, produce a *short commitment* c to a long, pseudorandom hidden-bit string ρ, as well as *openings* π_i to all the bits ρ_i of ρ. Then, VerifyBit takes as input the CRS, a short commitment, a position i, a value ρ_i, and an opening certificate π_i, and returns 0 or 1 depending on whether the opening is accepted. A dual-mode HBG must satisfy three properties:

- (Mode indistinguishability) the CRS can be generated in one of two modes, the *hiding* and the *binding* modes, which are computationally indistinguishable.
- (Hiding) when the CRS is in hiding mode, the value ρ_i at all non-opened positions i is *statistically hidden*, even given c and openings (ρ_j, π_j) at all other positions.

– (Extractable) when the CRS is in binding mode, there exists an efficient extractor which can extract from c a string ρ such that no efficient prover can produce accepting openings for $1 - \rho_i$, for any position i.

As shown in [34], and following related transformations in [9, 31, 41], a dual-mode HBG can be used to convert a NIZK for NP in the hidden-bits model (which exists unconditionally) into a *dual-mode* NIZK for NP in the CRS model (with statistical zero-knowledge when the HBG is used in hiding mode, and statistical soundness otherwise). These compilation techniques have their roots in the seminal works of Feige, Lapidot, and Shamir [16] and of Dwork and Naor [13].

Interactive Hidden-Bits Generators. The statistical NIZKs by Libert et al. [34] crucially rely on the dual-mode feature of the HBG: the statistical binding property appears unavoidable to compile a NIZK in the hidden-bits model. Hence, obtaining statistical zero-knowledge is done by generating the CRS in hiding mode, but switching it to the binding mode when analyzing soundness. Of course, this standard technique is limited to the CRS model.

In an exciting recent work [30], Kalai, Khurana, and Sahai, building upon previous results and ideas from [2,28,32], introduced an elegant and clever approach to partially emulate this "dual-mode feature" of the CRS model, but in the plain model. At a high level, they rely on statistically-hiding commitment schemes, which have the property that with some (negligible but *not too small*) probability, they will become binding and extractable; furthermore, this event cannot be detected by the committer. This in turn allows to obtain statistical privacy (e.g. statistical witness indistinguishability), while allowing to use the extractability properties to show soundness, at the cost of having to rely on assumptions which rule out even inverse-superpolynomial distinguishing advantages. This approach proved fruitful and led to a successful line of work [1, 22, 36] on building statistical ZAPs in the plain model.

Intuitively, our notion of interactive hidden-bits generator simply adapts this technique to the notion of dual-mode hidden-bits generator. That is, an IHBG is a pair $(\mathsf{GenBits}, \mathsf{VerifyBit})$, similar to a dual-mode HBG, with the following core differences:

– $\mathsf{GenBits}$ takes as input a uniformly random string, which will correspond to the verifier message in the ZAP.
– The non-opened values remain statistically hidden with overwhelming probability over the coins of $\mathsf{VerifyBit}$, for any (possibly malicious) choice of the random string.
– There exists a simulator which can produce simulated random coins (indistinguishable from true random coins) such that for any (possibly malicious) prover, with some not-too-small probability μ (e.g. inverse-superpolynomial) over the coins of the simulator, the hidden bit string ρ can be extracted from c.

Defining IHBG and Statistical ZAPs for NP. The above is of course very informal. Formally defining an interactive hidden-bits generator requires some care. In particular, we observe that the definition of extractability for statistically hiding extractable commitments in [1, 22, 36] do not suffice in our setting. At a high level, this is because these definition roughly say the following: the event that the commitments become extractable happens with probability μ, and whenever this event happens, the extracted value are *guaranteed* to be correct.

However, this will not hold in our setting: given a tuple $(c, \{i, \pi_i\}_i)$ of a commitment and set of openings from a possibly malicious prover, the hidden-bit string ρ recovered by the extractor is correct if $\mathsf{VerifyBit}(c, i, 1 - \rho_i, \pi_i) = \bot$ for all the opened positions i. Unfortunately, we can only guarantee that this will hold with overwhelming probability in our concrete construction, and not with probability 1. It turns out that, when building statistical ZAPs for NP, this is a crucial issue: in the soundness game of the ZAP construction from IHBG, the challenger will want extraction to succeed with probability μ *even when conditioning on other checks being successful.* A *guaranteed* correctness of extraction (conditioned on extraction succeeding) would ensure that this is the case, but an *overwhelming* probability of correctness does not, since conditioning on other events could arbitrarily change this probability.

To work around this issue, we adopt an approach closer in spirit to the definition of [37]. We define μ-extractability as follows: an IHBG is μ-extractable if there exists an efficient simulator $\mathsf{SimCoin}$ and an efficient opener Open such that, for any PPT adversary $\mathcal{A}$ *and any PPT distinguisher* D, given simulated coins $(\widetilde{r}, \tau) \leftarrow_r \mathsf{SimCoin}$ (where τ is an associated trapdoor for the opener), and a tuple

$(c, S, \rho_S^*, \{\pi_i\}_s, \mathsf{st}) \leftarrow_r \mathcal{A}(\widetilde{r})$ where c is a short commitment, S is a set of positions, ρ_S^* are the values which $\mathcal{A}$ opens the position to, the π_i are certificates of correct openings, and st is an arbitrary state, and letting $\rho \leftarrow \mathsf{Open}(\widetilde{r}, c, \tau)$, the probability p_1 that $\mathsf{VerifyBit}(\widetilde{r}, c, i, 1-\rho_i, \pi_i)$ returns $\perp$ for all $i \in S$ *and at the same time* the distinguisher D, given st, outputs 1, should satisfy

$$p_1 \geq \mu(\lambda) \cdot (p_2 - \mathsf{negl}(\lambda)),$$

where p_2 is the probability of the same event *without* the check that the procedure $\mathsf{VerifyBit}(\widetilde{r}, c, i, 1-\rho_i, \pi_i)$ returns $\perp$ for all $i \in S$. That is, μ-extractability requires that *for any other efficient conditions that we were verifying*, the probability that these conditions are *still verified* and that simultaneously, extraction succeeded and produced a correct output, should not decrease by a factor more than μ compared to the initial probability. This strong security notion is the key to capture the intuition that the extraction should succeed with probability μ *essentially independently of everything else.*

Given this notion of μ-extractable IHBG, we provide a natural construction of statistical ZAP for NP, which follows the standard template of using the IHBG to compile an unconditional NIZK for NP in the hidden-bits model, and formally prove that the resulting construction is a ZAP.

Constructing IHBG. It remains to construct IHBG with a statistical hiding property, satisfying the strong μ-extractability notion defined above. The first natural idea is to rely on the construction of dual-mode HBG from [34], and to convert it into a plain model protocol by letting the verifier sample the CRS herself. However, this immediately runs into obstacles: nothing prevents the verifier from sampling the CRS in binding mode, breaking the statistical hiding property. To recover the statistical hiding property, we let the prover *tweak* the CRS sampled by the verifier in a way that simultaneously guarantee two things:

- With overwhelming probability over the coins of the prover, the tweaked CRS will be in hiding mode, yet
- The tweak comes from a superpolynomial-size set, and by successfully guessing the tweak in advance, a simulator can engineer the sampled CRS (in a way that is indistinguishable from sampling a CRS honestly) such that the tweaked CRS will be in *binding* mode.

To achieve these two features, we rely on an elegant linear-algebra trick. In order to explain the idea, we first recall the high-level template of the construction of dual-mode HBG described in [34]. Let m be the length of the hidden bit string. The LPWW construction works in a hard-discrete-log group $\mathbb{G}$ of order p with generator g. It has the following structure:

- The hiding CRS is $g^{\mathbf{A}}$, where $\mathbf{A}$ is a random *full-rank* matrix $\mathbf{A} \in \mathbb{Z}_p^{(m+1)\times(m+1)}$.
- The binding CRS is $g^{\mathbf{A}}$, where $\mathbf{A}$ is a random *rank-1* matrix in $\mathbb{Z}_p^{(m+1)\times(m+1)}$.

Under the DDH assumption, the two modes are indistinguishable. Let $\mathbf{a}_0, \cdots, \mathbf{a}_m$ denote the columns of $\mathbf{A}$. To provide a short commitment to a pseudorandom length-m hidden bit string, the prover picks a random length-$(m+1)$ vector $\mathbf{y}$, and computes $c = g^{\mathbf{y}^\top \cdot \mathbf{a}_0}$. Then, the i-th hidden bit is defined to be $\rho_i = \mathsf{HB}(g^{\mathbf{y}^\top \cdot \mathbf{a}_i})$, where $\mathsf{HB}(\cdot)$ is a hardcore bit function (e.g. *a la* Goldreich-Levin). Eventually, to prove correct opening of ρ_i, given the commitment c and the CRS $g^{\mathbf{A}}$, the prover reveals $c_i = g^{\mathbf{y}^\top \cdot \mathbf{a}_i}$ and uses a NIZK to demonstrate the existence of a vector $\mathbf{y}$ such that $c = g^{\mathbf{y}^\top \cdot \mathbf{a}_0}$ and $c_i = g^{\mathbf{y}^\top \cdot \mathbf{a}_i}$ (from now on, we will call this language the LPWW language, $\mathscr{L}_{\mathsf{LPWW}}$).

Observe that when the CRS is in binding mode, we have $\mathbf{a}_i = v_i \cdot \mathbf{a}_0$ for some value v_i (since $\mathbf{A}$ has rank 1), hence the above language becomes essentially a DDH language. Adapting existing statistical NIZKs for the DDH language suffices to guarantee extractability in binding mode. On the other hand, when the CRS is in hiding mode, where $\mathbf{A}$ has full rank, *any* number of openings (of which there is at most m) $g^{\mathbf{y}^\top \cdot \mathbf{a}_i}$ leak *statistically* no information about the unopened values (since $\mathbf{A}$ is of dimension $(m+1)\times(m+1)$). This is because for any possible choice of values for the unopened positions, there exists a unique vector $\mathbf{y}$ that coincides with all the opened and unopened values when $\mathbf{A}$ is full rank. Hence, this guarantees statistical hiding.

Now, the core idea to achieve statistical hiding and μ-extractability in our construction (where μ is some arbitrary fixed inverse-superpolynomial function) is to let the verifier sample and send $g^{\mathbf{A}}$ herself, but to let the prover *tweak* this sample as follows: let $\mathbf{I}_{m+1}$ denote the identity matrix in $\mathbb{Z}_p^{(m+1)\times(m+1)}$. The prover picks a small exponent α at random from a subset of $\mathbb{Z}_p$ of size $\approx 1/\mu$,

e.g. by picking α as a random integer smaller than $[1/\mu]$, and using a natural encoding of integers in $\{0, \cdots, p-1\}$ as elements of $\mathbb{Z}_p$. Then, the prover defines the tweaked CRS $g^{\mathbf{A}'}$ to be $g^{\mathbf{A}-\alpha\cdot\mathbf{I}_{m+1}}$, and uses this tweaked CRS in the dual-mode HBG construction of [34].[6]

To see why this tweak achieves exactly what we want, observe that the following holds:

- First, we show that with overwhelming probability $1-(m+1)\mu$, the matrix $\mathbf{A}'$ has full rank. Indeed, if $\mathbf{A}'$ does *not* have full rank, it means that there is a nonzero vector $\mathbf{u}$ in the kernel of $\mathbf{A}'$. But then, $\mathbf{u}\cdot\mathbf{A}'=\mathbf{0}$ rewrites to $\mathbf{u}\cdot\mathbf{A}=\alpha\cdot\mathbf{u}$ – in equivalent terms, this means that α must be an *eigenvalue* of $\mathbf{A}$. But since $\mathbf{A}$ can have at most $m+1$ eigenvalues and α is randomly sampled from a set of size $1/\mu$, then this event can happen with probability at most $(m+1)\mu$.
- Second, we sketch why μ-extractability holds. First, the simulator will guess a value α', and set $\mathbf{A}\leftarrow\mathbf{M}+\alpha'\cdot\mathbf{I}_{m+1}$, where $\mathbf{M}$ is a rank-1 matrix. Observe that when the simulator guesses correctly, which happens with probability μ, it holds that $g^{\mathbf{A}'}$ is a binding CRS. Furthermore, under the assumption that no PPT adversary can distinguish DDH tuples from random tuples with probability better than $\mu\cdot\mathsf{negl}(\lambda)$, the replacement of truly random coins by simulated coins will not be detected. Hence, when further assuming that the ZAP for $\mathscr{L}_{\mathsf{LPWW}}$ guarantees a bound $\mu\cdot\mathsf{negl}(\lambda)$ on the probability that a malicious PPT prover breaks soundness, we can extract with probability almost μ a correct hidden-bit string. In Section 3, we will formally prove that μ-extractability holds with respect to an arbitrary PPT distinguisher D.

Summing up, the above provides a construction of IHBG (which in turns implies statistical ZAPs for NP), assuming

- the hardness of DDH with distinguishing advantage $\mu\cdot\mathsf{negl}(\lambda)$ for any PPT adversary and for any negligible functions μ and negl (an assumption in-between standard polynomial time hardness and superpolynomial time hardness, which is called *explicit hardness* in [1]), and
- the existence of statistical ZAPs for $\mathscr{L}_{\mathsf{LPWW}}$ with $\mu\cdot\mathsf{negl}(\lambda)$-soundness.

Instantiating the Statistical ZAPs for $\mathscr{L}_{\mathsf{LPWW}}$. Looking ahead, the formal analysis of our construction actually requires a slightly exotic notion of soundness: $\mathscr{L}_{\mathsf{LPWW}}$ is formally not a language, but a parametrized family of languages, and (adaptive) soundness must hold for parameters sampled uniformly at random from a specific subset of language parameters (which are those that correspond to $\mathbf{A}$ being of rank 1). We call a ZAP for the parameterized family of languages $\mathscr{L}_{\mathsf{LPWW}}$ IHBG-*friendly* when it satisfies this notion of soundness. We provide two instantiations for the underlying IHBG-friendly statistical ZAP.

Using pairings. First, we observe that the recent work of Couteau and Hartmann [8] provides a statistical ZAP for the DDH language, which extends directly to an IHBG-friendly statistical ZAP for the $\mathscr{L}_{\mathsf{LPWW}}$ language, under the standard kernel-DH assumption, in groups equipped with an asymmetric pairing. This leads to a statistical ZAP for NP under the explicit hardness of DDH in $\mathbb{G}_1$, and the explicit hardness of kernel-DH in $\mathbb{G}_2$, where $(\mathbb{G}_1,\mathbb{G}_2)$ are groups equipped with an asymmetric pairing.

Without pairings. Secondly, we revisit the recent construction of statistical NIZKs for the DDH language in pairing-free groups by Couteau, Katsumata, and Ursu [10]. Their construction relies on the assumption that no PPT algorithm can break the one-wayness of ElGamal against key-dependent message (OW-KDM) attacks with respect to efficient functions (i.e., the assumption that no PPT adversary can recover m from an ElGamal encryption of m, even when m is some efficiently computable function of the ElGamal secret key) with probability better than $2^{-3\lambda/4+o(\lambda)}$ (note that the best known PPT attack against this assumption, in appropriate groups, succeeds with probability $2^{-\lambda+o(\lambda)}$; furthermore, the restriction of KDM hardness to *efficient* functions of the secret key makes the assumption falsifiable "in spirit" – i.e., up to the negligible winning advantage). We denote this assumption the $2^{-3\lambda/4}$-OW-KDM hardness of ElGamal. We adapt the CKU construction to the LPWW language. Along the way, we put forth a modification of their construction which significantly improves the underlying assumption: we only need to assume that no PPT adversary can break the OW-KDM

[6] There is an obvious additional necessary change: when proving correctness of an opening, the statistical NIZK for $\mathscr{L}_{\mathsf{LPWW}}$ is replaced by a statistical ZAP for $\mathscr{L}_{\mathsf{LPWW}}$.

hardness of ElGamal with probability better than $2^{-\lambda/2+o(\lambda)}$. This change directly improves the result of [10]. With this instantiation, and observing that this statistical NIZK is also a statistical ZAP when the verifier can choose the CRS, we obtain a statistical ZAP for NP in pairing-free groups under the explicit hardness of DDH, and the $2^{-\lambda/2}$-OW-KDM hardness of ElGamal (we note that the latter is incomparable to DDH: it is a search, discrete-logarithm-type assumption, which is not even known to imply public-key encryption).

1.3 A Direct Construction using Pairings

Eventually, we point out that if one is willing to rely on a stronger assumption, one of our two instantiations (the pairing-based instantiation) can be obtained from our techniques in a much more direct (and simple-in-hindsight) way, without going through the hidden-bit model. Specifically, the core idea for our IHBG construction is to modify the CRS of a dual-mode NIZK using a simple tweak, sampled from a small set by the prover, which guarantees that with overwhelming probability a maliciously sampled CRS will be in hiding mode (but it will be in binding mode in the case when the verifier guesses the tweak).

A similar tweak can be applied directly to the dual-mode NIZK of Groth, Ostrovsky, and Sahai [24] instantiated with Groth-Sahai commitments [25]. Briefly, a Groth-Sahai commitment is of the form $(1, g^m) \cdot \mathbf{u}^r \cdot \mathbf{v}^s$, where $\mathbf{u}, \mathbf{v}$ are two random vectors of length two, and $\cdot$ denotes the coordinate-wise product (we write $\mathbf{u}^r$ for (u_1^r, u_2^r), where $\mathbf{u} = (u_1, u_2)$). When the vectors $(\mathbf{u}, \mathbf{v})$ are random, the commitments are perfectly hiding; when $\mathbf{v}$ is in the span of $\mathbf{u}$, they become perfectly binding. A GOS proof for circuit satisfiability, given a circuit C and a witness w such that $C(w) = 1$, works by committing to all bits of w, as well as to the bits on all wires during the evaluation of $C(w)$. Then, the proof proceeds by showing that all commitments commit to bits, that all gate relations are satisfied (which reduces to proving that a linear combination of the committed input and output bits – homomorphically computed from the commitments – is itself a bit), and that the output commitment contains 1. All these proofs can be reduced to pairing-product equations, hence can be proven with a Groth-Sahai NIZK [25].

Now, letting the verifier choose the CRS $(\mathbf{u}, \mathbf{v})$ themself, the prover can sample a small tweak $z \leftarrow_r [1/\mu]$, and set the CRS to be $(\mathbf{u}', \mathbf{v}') = (\mathbf{u} \cdot (1, g^z), \mathbf{v} \cdot (1, g^z))$. For any adversarial choice of $(\mathbf{u}, \mathbf{v})$, $(\mathbf{u}', \mathbf{v}')$ will not be colinear except with negligible property; on the other hand, with probability μ, the verifier can guess the tweak z and cause $(\mathbf{u}', \mathbf{v}')$ to be in binding mode. To make the analysis work, we need to rely on the same notion of μ-extractability which we defined previously. This direct approach leads to a statistical ZAP for NP in groups $(\mathbb{G}_1, \mathbb{G}_2)$ equipped with an asymmetric pairing, assuming the explicit hardness of DDH in both $\mathbb{G}_1$ and $\mathbb{G}_2$, a slightly stronger assumption compared to the one we obtain when going through the hidden-bit model. While simple in hindsight, this construction was apparently missed in previous works: the recent work of [37] achieved, under the same assumption, a strictly weaker result (a ZAPR argument for NP), using a considerably more involved and highly non-trivial construction.

2 Preliminaries

Notation. For integers $n \leq m$, we write $[m]$ for the set $\{1, \cdots, m\}$ and $[n : m]$ for the set $\{n, n+1, \cdots, m\}$. With an abuse of notation, for $r \in \mathbb{R}_{\geq 0}$, we use $[r]$ to denote $[\lceil r \rceil]$. For a string $x \in \{0,1\}^m$, set $S \subseteq [m]$, and an integer $i \in [m]$, denote x_S as the subsequence of the bits of x indexed by the set S and x_i as the i-th bit of x. We use bold fonts to denote column vectors over $\mathbb{Z}_p$ or $\mathbb{G}$ such as $\mathbf{v} = (v_1, \cdots, v^n)^\top \in \mathbb{Z}_p^n$ and $\mathbf{g} = (g^{a_1}, \cdots, g^{a_n})^\top \in \mathbb{G}^n$, respectively. We use capital bold fonts for matrices, e.g. $\mathbf{M}$. Given a vector $\mathbf{a}$, $g^{\mathbf{a}}$ denotes the column vector whose components are the $g^{a_i} \in \mathbb{G}$; we extend this notation to matrices of group elements $g^{\mathbf{M}}$ in the natural way. With an abuse of notation, we define $(\mathbf{g}^\top)^{\mathbf{v}}$ to be the inner-product between the exponent of $\mathbf{g}$ and $\mathbf{v}$, i.e., $g^{\sum_{i\in[n]} a_i v_i}$. For two vectors $\mathbf{v}$ and $\mathbf{w}$, $(\mathbf{v}|\mathbf{w})$ denotes a matrix with columns $\mathbf{v}$ and $\mathbf{w}$, and $(\mathbf{v}\|\mathbf{w})$ denotes a vector with $\mathbf{v}$ stacked on top of $\mathbf{w}$. Moreover, let $\mathsf{ker}(\mathbf{v})$ denote the linear subspace of $\mathbb{Z}_p^n$ consisting of all vectors $\mathbf{w}$ satisfying $\mathbf{v}^\top \mathbf{w} = 0$.

2.1 Hash Functions

We recall the definition of universal hash functions and define uniformity [26].

Definition 1 (Universal Hash Function). *An ensemble of a collection of hash functions* $\mathcal{H} = \{\mathcal{H}_\lambda\}_\lambda = \{\{H : X_\lambda \mapsto Y_\lambda\}_H\}_\lambda$ *is* universal *if for any* $x_0, x_1 \in X_\lambda$ *such that* $x_0 \neq x_1$*, we have* $\Pr[H \leftarrow_r \mathcal{H}_\lambda : H(x_0) = H(x_1)] \leq 1/|Y_\lambda|$.

Lemma 2 (Uniformity). *Let an ensemble of a collection of hash functions* $\mathcal{H} = \{\mathcal{H}_\lambda\}_\lambda = \{\{H : X_\lambda \mapsto Y_\lambda\}_H\}_\lambda$ *be* universal*. Then, for all* $\lambda \in \mathbb{N}$*, if* $|X_\lambda| \geq |Y_\lambda| \cdot \lambda^{\omega(1)}$*, the two distributions* $\{H \leftarrow_r \mathcal{H}_\lambda, x \leftarrow_r X_\lambda : (H, H(x))\}$ *and* $\{H \leftarrow_r \mathcal{H}_\lambda, y \leftarrow_r Y_\lambda : (H, y)\}$ *are* $\mathsf{negl}(\lambda)$*-statistically close.*

2.2 Hardness Assumptions

Let DHGen be a deterministic algorithm that on input 1^λ returns a description $\mathcal{G} = (\mathbb{G}, p)$ where $\mathbb{G}$ is a cyclic group of prime order p. Let PGen be a deterministic algorithm that on input 1^λ returns a description $\mathcal{PG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathsf{T}, p)$ where $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathsf{T})$ are cyclic groups of prime order p equipped with a bilinear pairing operation $\bullet : \mathbb{G}_1 \times \mathbb{G}_2 \mapsto \mathbb{G}_\mathsf{T}$. Below, we recall the definition of the decision Diffie-Hellman assumption in a cyclic group, as well as the definition of the kernel Diffie-Hellman assumption in a pairing group. Following [1], we also consider the *explicit* hardness of the assumptions, where we say that an assumption has explicit μ-hardness if μ is an explicit bound on the advantage of any polynomial time adversary. Note that this notion of explicit hardness is stronger than standard polynomial hardness, but weaker than superpolynomial hardness[7] for any superpolynomial factor.

Definition 3 (DDH Assumption). *We say that the decisional Diffie-Hellman (DDH) assumption holds relative to* DHGen *if for all PPT adversaries* $\mathcal{A}$*, it holds that* $\mathsf{Adv}^{\mathsf{DDH}}(\mathcal{A}) \leq \mathsf{negl}(\lambda)$*, where*

$$\mathsf{Adv}^{\mathsf{DDH}}(\mathcal{A}) = |\Pr\left[1 \leftarrow \mathcal{A}(1^\lambda, \mathcal{G}, g, g^\alpha, g^\beta, g^\gamma)\right] - \Pr\left[1 \leftarrow \mathcal{A}(1^\lambda, \mathcal{G}, g, g^\alpha, g^\beta, g^{\alpha\beta})\right]|.$$

Here, note that $\mathcal{G} \leftarrow \mathsf{DHGen}(1^\lambda)$ *and* DHGen *outputs a fixed group* $\mathbb{G}$ *per security parameter, and* $g \leftarrow_r \mathbb{G}$*,* $\alpha, \beta, \gamma \leftarrow_r \mathbb{Z}_p$ *are chosen uniformly. Furthermore, let* $\mu(\lambda)$ *be an efficiently computable function. We say that the* μ-explicit hardness of the DDH assumption *holds relative to* DHGen*, if* $\mathsf{Adv}^{\mathsf{DDH}}(\mathcal{A}) \leq \mu(\lambda)$ *for all PPT adversaries* $\mathcal{A}$*.*

We now recall the definition of the kernel Diffie-Hellman assumption in a pairing group. The kernel DH assumption is a standard search assumption in bilinear groups, introduced in [38] and used in several papers, e.g. [33]. In particular, kernel Diffie-Hellman in a group $\mathbb{G}_2$ is implied by (and is qualitatively weaker than) the DDH assumption in the same group.

Definition 4 (Kernel DH Assumption). *We say that the kernel Diffie-Hellman (kerDH) assumption holds relative to* PGen *if for all PPT adversaries* $\mathcal{A}$*, it holds that* $\mathsf{Adv}^{\mathsf{kerDH}}(\mathcal{A}) \leq \mathsf{negl}(\lambda)$*, where*

$$\mathsf{Adv}^{\mathsf{kerDH}}(\mathcal{A}) = \Pr\left[\begin{array}{l} \mathcal{PG} \leftarrow \mathsf{PGen}(1^\lambda), \\ (g_1, g_2) \leftarrow_r \mathbb{G}_1 \times \mathbb{G}_2, e \leftarrow_r \mathbb{Z}_p, \ : \ (u, v) \in \mathsf{ker}((1, e)^\top) \wedge v \neq 0 \\ (g_1^u, g_1^v) \leftarrow \mathcal{A}(1^\lambda, \mathcal{PG}, g_1, g_2, g_2^e) \end{array}\right].$$

Furthermore, let $\mu(\lambda)$ *be an efficiently computable function. We say that the* μ-explicit hardness of the kernel DH assumption *holds relative to* PGen*, if* $\mathsf{Adv}^{\mathsf{DDH}}(\mathcal{A}) \leq \mu(\lambda)$ *for all PPT adversaries* $\mathcal{A}$*.*

To see why the above is implied by DDH in $\mathbb{G}_2$, observe that on input $(g, g^\alpha, g^\beta, g^\gamma)$, an adversary against DDH can run the kernel DH adversary on input (g_1, g, g^α), where $g_1 \leftarrow_r \mathbb{G}_1$ and e is implicitly set as α. It then gets a vector (g_1^u, g_1^v) in $\mathbb{G}_1^2$ from the kernel DH adversary such that (u, v) is in the kernel of $(1, \alpha)$. Now, if $(g, g^\alpha, g^\beta, g^\gamma)$ is a DDH tuple, then (u, v) is also in the kernel of $(g^\beta, g^\gamma) = (g, g^\alpha)^\beta$, and this can be checked efficiently given (g_1^u, g_1^v) with the help of the pairing operation.

Remark 5 (Extensions to Matrix Diffie-Hellman). For the sake of concreteness and simplicity, we state our results in this paper in terms of the DDH and kernel DH assumptions. However, all our results can be generalized to hold under the standard generalizations of the Diffie-Hellman assumption, namely the decisional k-Lin [27] and kernel k-Lin assumptions, or even more generally any assumption from the family of the (decisional or kernel) matrix Diffie-Hellman assumptions [15, 38].

[7] We consider adversaries that run in superpolynomial time in case of superpolynomial hardness.

One-Way KDM Security of ElGamal. The last hardness assumption we will use in this work states, in essence, that no PPT adversary can recover m given an ElGamal encryption of m, even when m might be an efficiently computable function of the ElGamal secret key, with probability significantly better than $2^{-c\cdot\lambda}$ for some constant $c < 1$ (where λ is the logarithm of the group size). Note that the best known attack against this falsifiable search assumption succeeds with probability $\mathsf{poly}(\lambda)/2^{\lambda}$. To formally introduce the assumption, we introduce a natural secret-key variant of ElGamal (which suffices for our construction and leads to a more conservative assumption compared to the public-key variant).

Definition 6 (Secret-Key ElGamal). *Let $\tilde{\mathbb{G}} = \{\tilde{\mathbb{G}}_\lambda\}_\lambda$ be an ensemble of groups where each group $\tilde{\mathbb{G}}_\lambda$ is of order q such that $\lceil \log q \rceil \approx \lambda$. The natural (secret-key) variant of additive ElGamal with message space $\mathbb{Z}_q$ consists of the following three PPT algorithms.*

- $\mathsf{Setup}(1^\lambda)$: *The setup algorithm outputs a public-parameter $\tilde{G} \leftarrow_r \tilde{\mathbb{G}}_\lambda$ and a secret key $k \leftarrow_r \mathbb{Z}_q$.*
- $\mathsf{Enc}_{\tilde{G}}(k, m)$: *The encryption algorithm samples $\tilde{R} \leftarrow_r \tilde{\mathbb{G}}$ and outputs a ciphertext $\tilde{\mathbf{C}} = (\tilde{R}, \tilde{R}^k \cdot \tilde{G}^m)$.*
- $\mathsf{HalfDec}(k, \tilde{\mathbf{C}})$: *The* half *decryption algorithm parses $\tilde{\mathbf{C}}$ as $(\tilde{C}_0, \tilde{C}_1)$ and outputs $\tilde{C}_1/\tilde{C}_0^k$.*

Throughout the paper, we omit the subscript when the meaning is clear. Note that the scheme does not allow for full decryption, but only for decryption "up to discrete logarithm": for every $(\tilde{G}, k, m)$, it holds that $\mathsf{HalfDec}(k, \mathsf{Enc}_{\tilde{G}}(k, m)) = \tilde{G}^m$. One important property of the scheme is that it enjoys the notion of *universality*. Informally, the notion claims that the ciphertexts are not associated with a specific key, but rather, could have been an output of *any* key.

Definition 7 (Universality). *For all $\lambda \in \mathbb{N}$, $\tilde{G} \in \tilde{\mathbb{G}}_\lambda$, and $k^* \in \mathbb{Z}_q$, the ciphertexts of ElGamal satisfies*

$$\{\tilde{\mathbf{C}} : (k, m) \leftarrow_r \mathbb{Z}_q^2, \tilde{\mathbf{C}} \leftarrow_r \mathsf{Enc}_{\tilde{G}}(k, m)\} = \{\tilde{\mathbf{C}} : m \leftarrow_r \mathbb{Z}_q, \tilde{\mathbf{C}} \leftarrow_r \mathsf{Enc}_{\tilde{G}}(k^*, m)\} = \mathcal{U}_{\tilde{\mathbb{G}}^2}.$$

Definition 8 (OW-KDM Security). *Let $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ be an ensemble of sets of functions where each $\mathcal{F}_\lambda = \{F_u\}_u$ is a family of (possibly randomized) efficiently-computable functions. We say that ElGamal satisfies (one-query) δ-hard* OW-KDM *security with respect to $\mathcal{F}$ if for every $F_u \in \mathcal{F}_\lambda$, superpolynomial function s, and every (non-uniform) PPT adversary $\mathcal{A}$, it holds that*

$$\Pr_{\substack{(\tilde{G},k)\leftarrow_r \tilde{\mathbb{G}}_\lambda \times \mathbb{Z}_q \\ m \leftarrow F_u(\tilde{G},k) \\ \tilde{\mathbf{C}} \leftarrow_r \mathsf{Enc}_{\tilde{G}}(k,m)}} [\mathcal{A}(\tilde{G}, \tilde{\mathbf{C}}) = m] \leq s(\lambda) \cdot \delta(\lambda).$$

When ElGamal satisfies δ-hard OW-KDM *security for $\delta(\lambda) = 2^{-(c+o(1))\cdot\lambda}$ for some constant $c \in (0, 1]$, we say it is $2^{-c\lambda}$-*OW-KDM *secure or more simply, strong* OW-KDM *secure.*

The strong OW-KDM security of ElGamal was introduced in [6]. However, this work considered an extreme variant of the notion with $c = 1$ (that is, $2^{-\lambda}$-OW-KDM), and where security was required to hold with respect to *all functions* (even inefficient ones). The more conservative variant (with $c < 1$ and a restriction to efficiently computable functions) was introduced in [10], which used it (with constant $c = 3/4$) to build correlation-intractable hash functions. In this work, we will rely on an even more conservative variant with $c = 1/2$.

2.3 ZAP

ZAP [13, 14] is a public-coin two-move witness indistinguishable non-interactive argument. In this work, we focus on statistical ZAPs where witness indistinguishability holds unconditionally.

Definition 9 (ZAP). *A ZAP system Π_{ZAP} for an* NP *language $\mathscr{L} = \{\mathscr{L}_\lambda\}_\lambda$ with corresponding relation $\mathcal{R} = \{\mathcal{R}_\lambda\}_\lambda$ with public-coin length $\ell(\lambda)$ is a tuple of PPT algorithms* $(\mathsf{Prove}, \mathsf{Verify})$ *defined as follows.*

$\mathsf{Prove}(r, x, w) \to \pi$: *The proving algorithm is given the public-coin $r \in \{0,1\}^\ell$, a statement x, and a witness w, and outputs a proof π.*

$\mathsf{Verify}(r, x, \pi) \to \top$ **or** $\bot$: *The verification algorithm is given the public-coin* $r \in \{0,1\}^\ell$*, a statement* x*, and a proof* π*, and outputs* $\top$ *for acceptance or* $\bot$ *for rejection.*

We additionally require the following properties to hold.

Correctness: For any $\lambda \in \mathbb{N}$, $r \in \{0,1\}^\ell$ and $(x, w) \in \mathcal{R}_\lambda$, we have

$$\Pr[\mathsf{Verify}(r, x, \mathsf{Prove}(r, x, w)) = \top] = 1.$$

(Non-Adaptive) Computational Soundness: For any $\lambda \in \mathbb{N}$, PPT adversary $\mathcal{A}$, and any statement $x \notin \mathscr{L}_\lambda$, we have

$$\Pr[r \leftarrow \{0,1\}^\ell, \pi \leftarrow_r \mathcal{A}(r, x) : \mathsf{Verify}(r, x, \pi) = \top] \leq \mathsf{negl}(\lambda).$$

(Adaptive) Statistical Witness Indistinguishability: For any $\lambda \in \mathbb{N}$ and unbounded adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, we have

$$\left| \Pr\left[\begin{array}{l} (r, x, w_0, w_1, \mathsf{st}) \leftarrow_r \mathcal{A}_0(1^\lambda) \\ \pi_0 \leftarrow_r \mathsf{Prove}(r, x, w_0) \end{array} : \begin{array}{rl} & \mathcal{A}_1(\mathsf{st}, \pi_0) = 1 \\ \wedge & (x, w_0) \in \mathcal{R}_\lambda \\ \wedge & (x, w_1) \in \mathcal{R}_\lambda \end{array} \right] \right.$$
$$\left. - \Pr\left[\begin{array}{l} (r, x, w_0, w_1, \mathsf{st}) \leftarrow_r \mathcal{A}_0(1^\lambda) \\ \pi_1 \leftarrow_r \mathsf{Prove}(r, x, w_1) \end{array} : \begin{array}{rl} & \mathcal{A}_1(\mathsf{st}, \pi_1) = 1 \\ \wedge & (x, w_0) \in \mathcal{R}_\lambda \\ \wedge & (x, w_1) \in \mathcal{R}_\lambda \end{array} \right] \right| \leq \mathsf{negl}(\lambda).$$

Remark 10 (On Adaptive Soundness). In this work, we construct a ZAP that is *non-adaptive* computationally sound and adaptive *statistical* witness indistinguishable. This security property is in alignment with all the recent ZAPs (or ZAP with private randomness) [1, 22, 37]. Constructing ZAPs satisfying *adaptive* soundness and *statistical* witness indistinguishability seems to be difficult, where the former stipulates that the adversary can choose the statement $x \notin \mathscr{L}$ after it sees the public-coin r. Although we do not have any formal proofs of nonexistence of such ZAPs, we do have some evidence indicating the difficulty of obtaining them. In the context of NIZKs satisfying statistical zero-knowledge (NISZKs), Pass [40] shows that there is no black-box reduction from the adaptive soundness of NISZK to a falsifiable assumption [19, 39]. Owing to the similarity between ZAPs with statistical witness indistinguishability and NISZK, and the fact that the former turned out to be much more difficult to construct than the latter, we view it as an interesting open problem to construct adaptively sound proof systems.

2.4 NIZKs in the Hidden-Bits Model

We recall the notion of a NIZK in the hidden-bits model [17].

Definition 11. *A non-interactive proof system* Π_{HBM} *in the hidden-bits model for an* NP *language* $\mathscr{L} = \{\mathscr{L}_\lambda\}_\lambda$ *with corresponding relation* $\mathcal{R} = \{\mathcal{R}_\lambda\}_\lambda$ *with hidden-bits length* $m(\lambda)$ *is a pair of PPT algorithms* $(\mathsf{Prove}, \mathsf{Verify})$ *defined as follows.*

$\mathsf{Prove}(\mathsf{hb}, x, w) \to (I, \pi)$: *The proving algorithm is given a random bit string* $\mathsf{hb} \in \{0,1\}^m$ *and a statement* x*, and a witness* w *as inputs, and outputs a subset* $I \subseteq [m]$ *together with a proof* π*.*

$\mathsf{Verify}(S, \mathsf{hb}_S, x, \pi) \to \top$ **or** $\bot$: *The verification algorithm is given a subset* $S \subseteq [m]$*, a string* $\mathsf{hb}_S \in \{0,1\}^{|S|}$*, a statement* x *and a proof* π *as inputs, and outputs* $\top$ *for acceptance or* $\bot$ *for rejection.*

We additionally require the following properties to hold.

Correctness. For any $\lambda \in \mathbb{N}$, $(x, w) \in \mathcal{R}_\lambda$, any $\mathsf{hb} \in \{0,1\}^m$, and for $(I, \pi) \leftarrow_r \mathsf{Prove}(\mathsf{hb}, x, w)$, we have $\mathsf{Verify}(x, \mathsf{hb}_S, x, \pi) = \top$.

Statistical ε-Soundness. For any $\lambda \in \mathbb{N}$ and (possibly unbounded) adversary $\mathcal{A}$, we have

$$\Pr\left[\mathsf{hb} \leftarrow_r \{0,1\}^m, (x, S, \pi) \leftarrow_r \mathcal{A}(\mathsf{hb}) \ : \ \mathsf{Verify}(S, \mathsf{hb}_S, x, \pi) = \top \wedge x \notin \mathscr{L}_\lambda\right] \leq \varepsilon.$$

Perfect Zero-Knowledge. For any $\lambda \in \mathbb{N}$ and any (possibly unbounded) stateful adversary $\mathcal{A}$, there exists a PPT[8] zero-knowledge simulator Sim such that for every $(x, w) \in \mathcal{R}_\lambda$, the distributions $\{(S, \mathsf{hb}_S, \pi) : \mathsf{hb} \leftarrow_r \{0,1\}^m, (S, \pi) \leftarrow_r \mathsf{Prove}(\mathsf{hb}, x, w)\}$ and $\{\mathsf{Sim}_{\mathsf{zk}}(x)\}$ are perfectly indistinguishable.

We use the following result regarding the existence of NIZKs in the hidden-bits model [16].

[8] Note that we can also relax the definition to allow for an *unbounded* zero-knowledge simulator.

Theorem 12 (NIZK for all of NP in the hidden-bits model). *Let $k = k(\lambda)$ be any positive integer-valued function. Then, unconditionally, there exists a non-interactive proof system Π_{HBM} for any* NP *language $\mathscr{L} = \{\mathscr{L}_\lambda\}_\lambda$ in the hidden-bits model that uses $\mathsf{hb} = k \cdot \mathsf{poly}(\lambda)$ hidden-bits with soundness error $\epsilon \leq 2^{-k \cdot \lambda}$, where* poly *is a polynomial function related to the* NP *language $\mathscr{L}$.*

2.5 Correlation-Intractable Hash Functions

Finally, we recall the definition of correlation-intractable hash functions (CIH).

Definition 13 (Correlation Intractable Hash Function). *A collection $\mathcal{H} = \{H_\lambda : K_\lambda \times I_\lambda \mapsto O_\lambda\}_\lambda$ of (efficient) keyed hash functions is a $\mathcal{R}$-*correlation intractable hash *(CIH) family, with respect to a parameterized relation ensemble $\mathcal{R} = \{\mathcal{R}_\lambda\}_\lambda = \{\{\mathcal{R}_{\lambda,t} \subseteq I_\lambda \times O_\lambda\}_{t \in T_\lambda}\}_\lambda$, if for every (non-uniform) PPT adversary $\mathcal{A}$ and $t \in T_\lambda$, it holds that*

$$\Pr_{\substack{k \leftarrow_r K_\lambda \\ x \leftarrow_r \mathcal{A}(k)}} [(x, H_\lambda(k, x)) \in \mathcal{R}_{\lambda,t}] \leq \mathsf{negl}(\lambda).$$

Furthermore, let $\mu(\lambda)$ be an efficiently computable function. We say that the collection $\mathcal{H}$ satisfies $(\mu, \mathcal{R})$-correlation intractability *if the above probability is bounded by $\mu(\lambda)$ for all PPT adversaries $\mathcal{A}$.*

3 Interactive Hidden-Bits Generating Protocol and ZAPs for NP

In this section, we formally define an *interactive hidden-bits generating* (IHBG) protocol. Our definition builds on the definition of a (dual-mode) hidden-bits generator from [34, 41] (and the similar notion of (designated-verifier) PRG [9, 13, 14]). The main difference is that we allow a two-round interaction between the hidden-bits generator and the verifier, while removing the common reference string. Below, we define a public-coin flavor of an IHBG protocol to allow for public verifiability and reusability of the message from the verifier.

3.1 Definition

We formalize the notion of an interactive hidden-bits generating (IHBG) protocol.

Definition 14 (Interactive Hidden-Bits Generating Protocol). *Let $s(\lambda)$ and $m(\lambda)$ be positive valued polynomials. An interactive hidden-bits generating (*IHBG*) protocol Π_{IHBG} with public-coin length $\ell(\lambda)$ is a tuple of efficient algorithms* (GenBits, VerifyBit) *defined as follows.*

- $\mathsf{GenBits}(1^\lambda, m, r) \to (\sigma, \rho, \{\pi_i\}_{i \in [m]})$: *The hidden-bits generator algorithm is given the security parameter 1^λ (in unary), a length m, a public-coin $r \in \{0,1\}^\ell$ and outputs a commitment $\sigma \in \{0,1\}^s$, a string $\rho \in \{0,1\}^m$, and a set of proofs $\{\pi_i\}_{i \in m}$.*
- $\mathsf{VerifyBit}(r, \sigma, i, \rho_i, \pi_i) \to \top$ **or** $\bot$**:** *The verification algorithm is given a public-coin $r \in \{0,1\}^\ell$, a commitment $\sigma \in \{0,1\}^s$, a bit $\rho_i \in \{0,1\}$, and a proof π_i, and outputs $\top$ for acceptance or $\bot$ for rejection.*

We additionally require the following properties to hold. Below, we assume that the security parameter is provided to all algorithms, and omit it for simplicity.

Correctness: For any $\lambda \in \mathbb{N}$, $j \in [m]$, and $r \in \{0,1\}^\ell$, we have

$$\Pr[(\sigma, \rho, \{\pi_i\})_{i \in [m]} \leftarrow_r \mathsf{GenBits}(m, r) : \mathsf{VerifyBit}(r, \sigma, j, \rho_j, \pi_j) = \top] = 1.$$

Succinctness: The commitment length s only depends on the security parameter, i.e., $s(\lambda) = \mathsf{poly}(\lambda)$, and in particular, does not depend on the length m of the generated bits.

μ-Extractability: There exists a PPT public-coin simulator SimCoin and a deterministic polynomial-time open algorithm Open such that for all polynomial m, the following two conditions hold. For an intuitive explanation for μ-successful extraction, we refer the readers to the technical overview in Section 1.2.

– (Public-Coin Indistinguishability) for any PPT adversary $\mathcal{A}$, we have

$$\begin{aligned}&|\Pr[r \leftarrow_r \{0,1\}^\ell : \mathcal{A}(m,r)=1]\\ &-\Pr[(\widetilde{r},\tau)\leftarrow_r \mathsf{SimCoin}(1^\lambda,m) : \mathcal{A}(m,\widetilde{r})=1]| \le \mathsf{negl}(\lambda).\end{aligned}$$

– (μ-Successful Extraction) for any PPT adversary $\mathcal{A}$ and any PPT distinguisher D, we have

$$\begin{aligned}&\Pr\left[\begin{array}{l}(\widetilde{r},\tau)\leftarrow_r \mathsf{SimCoin}(1^\lambda,m)\\ (\sigma,S,\rho^*_S,\{\pi_i\}_{i\in S},\mathsf{st})\leftarrow_r \mathcal{A}(m,\widetilde{r}) :\\ \rho\leftarrow \mathsf{Open}(\widetilde{r},\sigma,\tau)\end{array}\begin{array}{c}D(\mathsf{st})=1 \ \wedge\ \rho\in\{0,1\}^m \ \wedge\ \forall i\in S,\\ \mathsf{VerifyBit}(\widetilde{r},\sigma,i,1-\rho_i,\pi_i)=\bot\end{array}\right]\\ &\ge \mu(\lambda)\cdot\Pr\left[\begin{array}{l}(\widetilde{r},\tau)\leftarrow_r \mathsf{SimCoin}(1^\lambda,m)\\ (\sigma,S,\rho^*_S,\{\pi_i\}_{i\in S},\mathsf{st})\leftarrow_r \mathcal{A}(m,\widetilde{r})\end{array} : \ D(\mathsf{st})=1\right]-\mu(\lambda)\cdot\mathsf{negl}(\lambda).\end{aligned}$$

Statistical Hiding: For all polynomial m, public-coin $r\in\{0,1\}^\ell$, and all unbounded adversaries $\mathcal{A}=(\mathcal{A}_0,\mathcal{A}_1)$, there exists a (possibly unbounded) simulator Sim such that

$$\begin{aligned}&\left|\Pr\left[\begin{array}{l}(\sigma,\rho,\{\pi_i\}_{i\in[m]})\leftarrow_r \mathsf{GenBits}(m,r)\\ S\leftarrow_r \mathcal{A}_0(\rho)\end{array} : S\subseteq[m] \ \wedge\ \mathcal{A}_1(r,S,\sigma,\rho,\{\pi_i\}_{i\in S})=1\right]-\right.\\ &\left.\Pr\left[\begin{array}{l}\rho\leftarrow_r\{0,1\}^m, S\leftarrow_r\mathcal{A}_0(\rho)\\ (\sigma,\{\pi_i\}_{i\in S})\leftarrow_r \mathsf{Sim}(m,r,S,\rho_S)\end{array} : S\subseteq[m] \ \wedge\ \mathcal{A}_1(r,S,\sigma,\rho,\{\pi_i\}_{i\in S})=1\right]\right|\\ &\le \mathsf{negl}(\lambda).\end{aligned}$$

3.2 ZAPs for NP from Interactive Hidden-Bits Generating Protocols

Here, we construct a ZAP for NP based on an IHBG protocol and a NIZK in the hidden-bits model, where the latter exists unconditionally.

Building Block. Let $\mathscr{L}$ be an NP language and $\mathcal{R}$ be its corresponding relation.[9] We construct a ZAP for $\mathscr{L}$ based on the following building blocks.

– $\Pi_{\mathsf{IHBG}}=(\mathsf{GenBits},\mathsf{VerifyBit})$ is an interactive hidden-bits generating protocol. We assume it has public-coin length $\ell(\lambda)$, commitment length $s(\lambda)$, and output length $m(\lambda)$ (i.e., $\rho\in\{0,1\}^m$). We further assume it satisfies $\mu(\lambda)$-extractability.
– $\Pi_{\mathsf{HBM}}=(\mathsf{HBM.Prove},\mathsf{HBM.Verify})$ is a NIZK in the hidden-bits model for $\mathscr{L}$. We assume the hidden-bits length is $m(\lambda)$ and it is statistically $\varepsilon_{\mathsf{HBM}}$-sound, where $\varepsilon_{\mathsf{HBM}}=2^{-s(\lambda)}\cdot\mu(\lambda)\cdot\mathsf{negl}(\lambda)$.[10]

Construction. The construction of a ZAP for $\mathscr{L}$ with public-coin length $\ell'(\lambda)=\ell(\lambda)+m(\lambda)$, denoted as Π_{ZAP}, is described as follows.

$\mathsf{ZAP.Prove}(r',x,w)$: On input a public-coin $r'\in\{0,1\}^{\ell'}$, a statement x and a witness w, parse it as $(r,\Delta)\leftarrow r'$ such that $r\in\{0,1\}^\ell$ and $\Delta\in\{0,1\}^m$. Then run $(\sigma,\rho,\{\pi_{\mathsf{IHBG},i}\}_{i\in[m]})\leftarrow_r \mathsf{GenBits}(1^\lambda,m,r)$ and compute an HBM proof $(S,\pi_{\mathsf{HBM}})\leftarrow_r \mathsf{HBM.Prove}(\mathsf{hb},x,w)$, where $\mathsf{hb}:=\rho\oplus\Delta$. Finally, output $\pi_{\mathsf{ZAP}}=(\sigma,S,\rho_S,\{\pi_{\mathsf{IHBG},i}\}_{i\in S},\pi_{\mathsf{HBM}})$.

$\mathsf{ZAP.Verify}(r',x,\pi_{\mathsf{ZAP}})$: On input a public-coin $r'\in\{0,1\}^{\ell'}$, a statement x and a proof π_{ZAP}, parse it as $(r,\Delta)\leftarrow r'$ such that $r\in\{0,1\}^\ell$ and $\Delta\in\{0,1\}^m$, and $(\sigma,S,\rho_S,\{\pi_{\mathsf{IHBG},i}\}_{i\in S},\pi_{\mathsf{HBM}})\leftarrow\pi_{\mathsf{ZAP}}$. Then, output $\top$ if $\mathsf{HBM.Verify}(S,\rho_S\oplus\Delta_S,x,\pi_{\mathsf{HBM}})=\top$ and $\mathsf{VerifyBit}(r,\sigma,i,\rho_i,\pi_{\mathsf{IHBG},i})=\top$ for all $i\in S$. Otherwise, output $\bot$.

3.3 Security

Correctness of our ZAP follows from a routine check. Below, we show our ZAP satisfies non-adaptive computational soundness and adaptive statistical witness indistinguishability in Theorems 15 and 19.

[9] Although $\mathscr{L}$ and $\mathcal{R}$ are parameterized by the security parameter λ, we omit them throughout the paper for better readability whenever the meaning is clear.

[10] Here, m can be set sufficiently large for both Π_{IHBG} and Π_{HBM} so that the existence of Π_{HBM} is guaranteed unconditionally by Theorem 12.

Theorem 15 (Soundness). *If Π_{IHBG} is μ-extractable and Π_{HBM} has statistical $\varepsilon_{\mathsf{HBM}}$-soundness, where $\varepsilon_{\mathsf{HBM}} = 2^{-s(\lambda)} \cdot \mu(\lambda) \cdot \mathsf{negl}(\lambda)$, then Π_{ZAP} has non-adaptive computational soundness.*

Proof. Assume there exists a statement $x \notin \mathscr{L}$ and a PPT adversary $\mathcal{A}$ against the non-adaptive computational soundness of Π_{ZAP} with advantage ε. Below, we consider the following sequence of games between $\mathcal{A}$ and a challenger and denote E_i as the event that the challenger outputs 1.

Game_1**:** This is the real soundness game that proceeds as follows: The challenger first samples a public-coin $r' \leftarrow_r \{0,1\}^{\ell'}$ and sends it to $\mathcal{A}$. $\mathcal{A}$ then outputs a proof π^*_{ZAP} and sends it to the challenger. The challenger outputs 1 if $\mathsf{ZAP.Verify}(r', x, \pi^*_{\mathsf{ZAP}}) = \top$, and outputs 0 otherwise. By definition $\Pr[\mathsf{E}_1] = \varepsilon$.

Game_2**:** This game is identical to the previous game except that the public-coin $r' \in \{0,1\}^{\ell'}$ is sampled differently. Let $\mathsf{SimCoin}$ be the PPT public-coin simulator of the IHBG protocol Π_{IHBG}. Then, in this game, the challenger first runs $(\widetilde{r}, \tau) \leftarrow_r \mathsf{SimCoin}(m)$ and samples $\Delta \leftarrow_r \{0,1\}^m$, where $\widetilde{r} \in \{0,1\}^{\ell}$, and outputs the simulated public-coin $\widetilde{r}' := (\widetilde{r}, \Delta) \in \{0,1\}^{\ell'}$. The rest is defined the same as in the previous game.

Game_3**:** This game is identical to the previous game except that the challenger checks an additional condition regarding π^*_{ZAP} output by $\mathcal{A}$. Let Open be the efficient deterministic open algorithm of the IHBG protocol Π_{IHBG}. Then, in this game, when $\mathcal{A}$ outputs π^*_{ZAP}, the challenger first parses

$$(\sigma^*, S^*, \rho^*_{S^*}, \{\pi^*_{\mathsf{IHBG},i}\}_{i \in S^*}, \pi^*_{\mathsf{HBM}}) \leftarrow \pi^*_{\mathsf{ZAP}}$$

and runs $\rho \leftarrow \mathsf{Open}(\widetilde{r}, \sigma^*, \tau)$. It then outputs 1 if $\mathsf{ZAP.Verify}(r', x, \pi^*_{\mathsf{ZAP}}) = \top$, $\rho \in \{0,1\}^m$, and $\rho^*_{S^*} = \rho_{S^*}$, and 0 otherwise.

The following Lemmas 16 to 18 establish $\Pr[\mathsf{E}_1] = \varepsilon \le \mathsf{negl}(\lambda)$, thus completing the proof.

Lemma 16. *If Π_{IHBG} is μ-extractable for all PPT adversary, then we have $|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_2]| \le \mathsf{negl}(\lambda)$, hence $\Pr[\mathsf{E}_2] \ge \varepsilon - \mathsf{negl}(\lambda)$.*

Proof. The only difference between the two games is how the public-coin is generated. Let us consider the following adversary $\mathcal{B}$ against the public-coin indistinguishability of Π_{IHBG}: $\mathcal{B}$ receives $r \in \{0,1\}^{\ell}$ from its challenger and samples $\Delta \leftarrow_r \{0,1\}^m$. It then invokes $\mathcal{A}$ on input $r' = (r, \Delta)$, and outputs 1 if the proof π_{ZAP} output by $\mathcal{A}$ satisfies $\mathsf{ZAP.Verify}(r', x, \pi^*_{\mathsf{ZAP}}) = \top$, and 0 otherwise. Since $\mathcal{B}$ perfectly simulates Game_1 (resp. Game_2) when $r \leftarrow_r \{0,1\}^{\ell}$ (resp. $(r, \tau) \leftarrow_r \mathsf{SimCoin}(m)$), we have $|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_2]| \le \mathsf{negl}(\lambda)$.

Lemma 17. *If Π_{IHBG} is μ-extractable for all PPT adversary, then we have $\Pr[\mathsf{E}_3] \ge \mu(\lambda) \cdot (\Pr[\mathsf{E}_2] - \mathsf{negl}(\lambda))$.*

Proof. This follows from the μ-successful extractability of Π_{IHBG}. Let us consider the following adversary $\mathcal{B}$ and distinguisher $\mathcal{D}$ against the μ-successful extractability: $\mathcal{B}$ on input m and $\widetilde{r}$ invokes $\mathcal{A}$ and simulates the challenger in Game_2. When $\mathcal{A}$ outputs a forgery $\pi^*_{\mathsf{ZAP}} = (\sigma^*, S^*, \rho^*_{S^*}, \{\pi^*_{\mathsf{IHBG},i}\}_{i \in S^*}, \pi^*_{\mathsf{HBM}})$, $\mathcal{B}$ outputs $(\sigma^*, S^*, \rho^*_{S^*}, \{\pi^*_{\mathsf{IHBG},i}\}_{i \in S^*}, \mathsf{st})$, where $\mathsf{st} = (\widetilde{r}, \pi^*_{\mathsf{ZAP}})$; $\mathcal{D}$ on input st, checks if $\mathsf{ZAP.Verify}(\widetilde{r}, x, \pi^*_{\mathsf{ZAP}}) = \top$, and outputs 1 if so and outputs 0 otherwise. Observe that the probability $\mathcal{D}$ outputs 1 is the same as the probability that event E_2 occurs. Below, we relate the probability that event E_3 occurs with the left hand side equation of μ-successful extractability.

The only difference between Game_2 and Game_3 is the check that $\rho \in \{0,1\}^m$ and $\rho^*_{S^*} = \rho_{S^*}$. Now, consider a variant Game'_3 of Game_3 where, instead of checking $\rho^*_{S^*} = \rho_{S^*}$, the challenger checks that for all $i \in S^*$, it holds that

$$\mathsf{VerifyBit}(\widetilde{r}, \sigma^*, i, 1 - \rho_i, \pi^*_{\mathsf{IHBG},i}) = \bot.$$

Let E'_3 be the event that the challenger outputs 1 in this variant. Observe that if event E'_3 occurs then so does event E_3. Indeed, whenever the challenger outputs 1 in E'_3, it holds in particular that

$$\forall i \in S^*,\ \mathsf{VerifyBit}(\widetilde{r}, \sigma^*, i, \rho^*_i, \pi^*_i) = \top, \text{ and}$$
$$\forall i \in S^*,\ \mathsf{VerifyBit}(\widetilde{r}, \sigma^*, i, 1 - \rho_i, \pi^*_i) = \bot.$$

The latter implies that it can never hold, for any $i \in S^*$, that $\rho^*_i = 1 - \rho_i$; hence, since we check $\rho \in \{0,1\}^m$ in both events, whenever E'_3 happens, it further holds that $\rho^*_{S^*} = \rho_{S^*}$ and E_3 therefore holds as well. In other terms,

$$\Pr[\mathsf{E}_3] \ge \Pr[\mathsf{E}'_3].$$

Therefore, by applying the μ-successful extractability of Π_{IHBG} with respect to $\mathcal{B}$ and $\mathcal{D}$, since the only difference between Game_2 and Game_3' is the check that $\rho \in \{0,1\}^m$ and $\mathsf{VerifyBit}(\widetilde{r}, \sigma^*, i, 1-\rho_i, \pi^*_{\mathsf{IHBG},i}) = \bot$, we get

$$\Pr[\mathsf{E}_3'] \geq \mu(\lambda) \cdot (\Pr[\mathsf{E}_2] - \mathsf{negl}(\lambda)),$$

which concludes the proof of Lemma 17.

Lemma 18. *If Π_{HBM} is statistical $\varepsilon_{\mathsf{HBM}}$-sound, then we have $\Pr[\mathsf{E}_3] \leq \mu(\lambda) \cdot \mathsf{negl}(\lambda)$.*

Proof. Let $(\sigma^*, S^*, \rho^*_{S^*}, \{\pi^*_{\mathsf{IHBG},i}\}_{i\in S^*}, \pi^*_{\mathsf{HBM}}) \leftarrow \pi^*_{\mathsf{ZAP}}$ be $\mathcal{A}$'s output. When the challenger outputs 1 (i.e., event E_3 occurs), we have $\rho^*_{S^*} = \rho_{S^*}$, where $\rho \leftarrow \mathsf{Open}(\widetilde{r}, \sigma^*, \tau)$, and $\mathsf{HBM.Verify}(S^*, \rho^*_{S^*} \oplus \Delta_{S^*}, x, \pi^*_{\mathsf{HBM}}) = \top$. For an any $S^* \subseteq [m]$ and ρ_{S^*}, if $\Delta \leftarrow_r \{0,1\}^m$ is sampled uniformly at random, then $\rho_{S^*} \oplus \Delta_{S^*}$ is distributed uniformly random. Then, by soundness of Π_{HBM}, for a fixed ρ_{S^*} we have

$$\Pr[\mathsf{HBM.Verify}(S^*, \rho_{S^*} \oplus \Delta_{S^*}, x, \pi^*_{\mathsf{HBM}}) = \top] \leq \varepsilon_{\mathsf{HBM}},$$

where the probability is taken over the randomness of Δ, $\mathcal{A}$, and the challenger, conditioned on $\mathcal{A}$ outputting $\rho^*_{S^*}$ that is consistent with ρ_{S^*}. Here, we do not include the condition $x \notin \mathscr{L}$ in the above equation since we consider non-adaptive soundness for Π_{ZAP}.

If we fix an arbitrary $(\widetilde{r}, \tau)$, then for any commitment $\sigma \in \{0,1\}^s$ the output of $\rho \leftarrow \mathsf{Open}(\widetilde{r}, \sigma, \tau)$ is uniquely defined since Open is deterministic. Let us denote the unique ρ as ρ^σ. Then, taking a union bound over all possible commitments $\sigma \in \{0,1\}^s$, we have

$$\begin{aligned}\Pr[\exists \sigma \in \{0,1\}^\ell \text{ s.t. } \mathsf{HBM.Verify}(S^*, \rho^\sigma_{S^*} \oplus \Delta_{S^*}, x, \pi^*_{\mathsf{HBM}}) = \top] &\leq 2^s \cdot \varepsilon_{\mathsf{HBM}} \\ &= \mu(\lambda) \cdot \mathsf{negl}(\lambda).\end{aligned}$$

Thus, we conclude $\Pr[\mathsf{E}_3] \leq \mu(\lambda) \cdot \mathsf{negl}(\lambda)$.

Putting everything together, this gives $\mu(\lambda)\cdot(\varepsilon - \mathsf{negl}(\lambda)) \leq \mu(\lambda)\cdot\mathsf{negl}(\lambda)$, which implies $\varepsilon \leq \mathsf{negl}(\lambda)$. This concludes the proof.

Theorem 19 (Statistical Witness Indistinguishability). *If Π_{IHBG} is statistically hiding and Π_{HBM} has perfect zero-knowledge, then Π_{ZAP} is adaptive statistical witness indistinguishability.*

Proof. We proceed with a sequence of games where the first (resp. last) corresponds to the game where the challenger uses witness w_0 (resp. w_1). Without loss of generality, we assume the adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ always outputs a statement x and witnesses w_0 and w_1 such that $(x, w_0) \in R$ and $(x, w_1) \in R$. We denote E_i as the event that $\mathcal{A}$ outputs 1 on input a proof π_{ZAP}.

Game_1**:** This is the real game where the challenger uses witness w_0. Concretely, the challenger runs $(r', x, w_0, w_1, \mathsf{st}) \leftarrow_r \mathcal{A}_0(1^\lambda)$ and runs $\pi_0 \leftarrow_r \mathsf{ZAP.Prove}(r', x, w_0)$. Here, recall $\mathsf{ZAP.Prove}$ constitutes of the following steps:

1. Parse $(r, \Delta) \leftarrow r'$;
2. Compute $(\sigma, \rho, \{\pi_{\mathsf{IHBG},i}\}_{i\in[m]}) \leftarrow_r \mathsf{GenBits}(m, r)$;
3. $(S, \pi_{\mathsf{HBM}}) \leftarrow_r \mathsf{HBM.Prove}(\mathsf{hb}, x, w_0)$, where $\mathsf{hb} := \rho \oplus \Delta$;
4. Output $\pi_{\mathsf{ZAP}} = (\sigma, S, \rho_S, \{\pi_{\mathsf{IHBG},i}\}_{i\in S}, \pi_{\mathsf{HBM}})$.

By definition, we have $\Pr[\mathcal{A}_1(\mathsf{st}, \pi_{\mathsf{ZAP}})] = \Pr[\mathsf{E}_0]$.

Game_2**:** We modify how the challenger computes the proof π_{ZAP}. It uses the (possibly unbounded) simulator $\mathsf{IHBG.Sim}$ guaranteed by the statistically hiding property of IHBG to simulate the proofs $\{\pi_{\mathsf{IHBG},i}\}_{i\in S}$ for the opening of the hidden-bits. Concretely, the challenger performs the following instead of running $\mathsf{ZAP.Prove}(r', x, w_0)$, where the red underline indicates the difference between the previous game.

1. Parse $(r, \Delta) \leftarrow r'$;
2. Sample $\rho \leftarrow_r \{0,1\}^m$;
3. $(S, \pi_{\mathsf{HBM}}) \leftarrow_r \mathsf{HBM.Prove}(\mathsf{hb}, x, w_0)$, where $\mathsf{hb} := \rho \oplus \Delta$;
4. $(\sigma, \{\pi_{\mathsf{IHBG},i}\}_{i\in S}) \leftarrow_r \mathsf{IHBG.Sim}(m, r, S, \rho_S)$;
5. Output $\pi_{\mathsf{ZAP}} = (\sigma, S, \rho_S, \{\pi_{\mathsf{IHBG},i}\}_{i\in S}, \pi_{\mathsf{HBM}})$.

Game_3: We further modify how the challenger computes the proof π_{ZAP}. It uses the zero-knowledge simulator $\mathsf{HBM.Sim}$ guaranteed by the perfect zero-knowledge of the NIZK in the hidden-bits model to compute the proof π_{HBM}. Concretely, the challenger performs the following, where the red underline indicates the difference between the previous game. Here, notice that the challenger no longer uses the witness w_0.

1. Parse $(r, \Delta) \leftarrow r'$;
2. $(S, \mathsf{hb}_S, \pi_{\mathsf{HBM}}) \leftarrow_r \mathsf{HBM.Sim}(x)$;
3. $\rho_S \leftarrow \mathsf{hb}_S \oplus \Delta_S$;
4. $(\sigma, \{\pi_{\mathsf{IHBG},i}\}_{i\in S}) \leftarrow_r \mathsf{IHBG.Sim}(m, r, S, \rho_S)$
5. Output $\pi_{\mathsf{ZAP}} = (\sigma, S, \rho_S, \{\pi_{\mathsf{IHBG},i}\}_{i\in S}, \pi_{\mathsf{HBM}})$.

Game_4: We undo the change we made to move from Game_2 to Game_3, except that the challenger uses witness w_1 instead of w_0 to compute π_{HBM} in Item 3 of Game_2.

Game_5: We undo the change we made to move from Game_1 to Game_2. This game is the real game where the challenger uses witness w_1.

The following two lemmas establish $|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_3]| \leq \mathsf{negl}(\lambda)$. Since the game transition from Game_3 to Game_5 is identical to those from Game_3 to Game_1 except that the challenger uses w_1 instead of w_0, we have $|\Pr[\mathsf{E}_3] - \Pr[\mathsf{E}_5]| \leq \mathsf{negl}(\lambda)$. Collecting the bounds, we obtain $|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_5]| \leq \mathsf{negl}(\lambda)$, thus completing the proof.

Lemma 20. *If Π_{IHBG} is statistically hiding, then we have $|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_2]| \leq \mathsf{negl}(\lambda)$.*

Proof. We construct an adversary $\mathcal{B}$ against the statistically hiding property of the IHBG protocol that simulates the view of $\mathcal{A}$ in either Game_1 or Game_2 depending on the challenge it receives. Since statistical hiding is defined with respect to any public-coin $r \in \{0,1\}^\ell$, we can think $\mathcal{B}$ prepares r before seeing the hidden-bits $\rho \in \{0,1\}^m$. For clarity, we assume $\mathcal{B}$ consists of three unbounded algorithms $(\mathcal{B}'_0, \mathcal{B}_0, \mathcal{B}_1)$, where $\mathcal{B}'_0$ prepares r and $(\mathcal{B}_0, \mathcal{B}_1)$ are as defined in Definition 14.

Description of $\mathcal{B}$ follows: $\mathcal{B}'_0$ runs $(r', x, w_0, w_1, \mathsf{st}) \leftarrow \mathcal{A}_0(1^\lambda)$ and parses $(r, \Delta) \leftarrow r'$. It then outputs the public-coin $r \in \{0,1\}^\ell$ to the challenger. The challenger either runs (I) $(\sigma, \rho, \{\pi_{\mathsf{IHBG},i}\}_{i\in[m]}) \leftarrow_r \mathsf{GenBits}(m, r)$ or (II) $\rho \leftarrow_r \{0,1\}^m$ and sends ρ to $\mathcal{B}_0$. $\mathcal{B}_0$ then sets $\mathsf{hb} := \rho \oplus \Delta$, runs $(S, \pi_{\mathsf{HBM}}) \leftarrow_r \mathsf{HBM.Prove}(\mathsf{hb}, x, w_0)$, and sends the index set $S \in [m]$ to the challenger. The challenger then provides $\mathcal{B}_1$ with input $(r, S, \sigma, \rho, \{\pi_{\mathsf{IHBG},i}\}_{i\in S})$. If the challenger ran (I) above, then it uses the already generated $(\sigma, \{\pi_{\mathsf{IHBG},i}\}_{i\in S})$ and if the challenger ran (II) above, then it uses $(\sigma, \{\pi_{\mathsf{IHBG},i}\}_{i\in S}) \leftarrow_r \mathsf{IHBG.Sim}(m, r, S, \rho_S)$. $\mathcal{B}_1$ then prepares the proof $\pi_{\mathsf{ZAP}} = (\sigma, S, \rho_S, \{\pi_{\mathsf{IHBG},i}\}_{i\in S}, \pi_{\mathsf{HBM}})$ and outputs whatever output by $\mathcal{A}_1(\mathsf{st}, \pi_{\mathsf{ZAP}})$.

It can be checked that $\mathcal{B}$ perfectly simulates the view of Game_1 (resp. Game_2) to $\mathcal{A}$ when the challenger runs (I) (resp. (II)). Therefore, assuming statistical hiding of the IHBG protocol, we have $|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_2]| \leq \mathsf{negl}(\lambda)$ as desired.

Lemma 21. *If Π_{HBM} has perfect zero-knowledge, then we have $\Pr[\mathsf{E}_2] = \Pr[\mathsf{E}_3]$.*

Proof. Fix an arbitrary $\Delta \in \{0,1\}^m$. Then, in Game_2, we have $(S, \pi_{\mathsf{HBM}}) \leftarrow_r \mathsf{HBM.Prove}(\mathsf{hb}, x, w_0)$ for a uniform random hb since $\mathsf{hb} := \rho \oplus \Delta$ for a uniform random $\rho \leftarrow_r \{0,1\}^m$. On the other hand, in Game_3, we have $(S, \mathsf{hb}_S, \pi_{\mathsf{HBM}}) \leftarrow_r \mathsf{HBM.Sim}(x)$ and $\rho_S := \mathsf{hb}_S \oplus \Delta_S$. By the perfect zero-knowledge of Π_{HBM}, the distributions of $(S, \rho_S, \pi_{\mathsf{HBM}})$ are identical. Moreover, $(\sigma, \{\pi_{\mathsf{IHBG},i}\}_{i\in S})$ are generated identically in both games. Therefore, the distribution of $\pi_{\mathsf{ZAP}} = (\sigma, S, \rho_S, \{\pi_{\mathsf{IHBG},i}\}_{i\in S}, \pi_{\mathsf{HBM}})$ is identical in both games. Thus, we have $\Pr[\mathsf{E}_2] = \Pr[\mathsf{E}_3]$ as desired.

4 The LPWW Language $\mathscr{L}_{\mathsf{LPWW}}$

To instantiate the generic construction of statistical ZAP for NP given in Section 3, we will construct an IHBG which builds upon the dual-mode hidden-bit generator of Libert, Passelègue, Wee, and Wu [34]. In this section, we first recall the specific *parameterized* language considered by [34] (denoted as the LPWW language $\mathscr{L}_{\mathsf{LPWW}}$). We then introduce some tools related to this parameterized language: a specific type of statistical ZAP for $\mathscr{L}_{\mathsf{LPWW}}$, which we call IHBG-*friendly statistical ZAP*, and a Σ-protocol for $\mathscr{L}_{\mathsf{LPWW}}$.

4.1 Definition

Formally, we denote by $\mathscr{L}_{\mathsf{LPWW}} := \{\mathscr{L}_{\mathsf{LPWW},\lambda}\}_\lambda$ the following family of parametrized languages: let $\mathbb{G}$ be a cyclic group of prime order p. We implicitly fix a vector length $d \in \mathbb{N}$ and a generator $g \in \mathbb{G}$ for each security parameter λ.[11] Let a set of parameter space Λ_λ be $(\mathbb{G}^d \backslash \{\mathbf{1}\})^2$, where $\mathbf{1} := g^{\mathbf{0}}$ for $\mathbf{0} \in \mathbb{Z}_p^d$. Then, for any parameter $\mathsf{par} = (g^{\mathbf{v}}, g^{\mathbf{w}}) \in \Lambda_\lambda$, we define $\mathscr{L}_{\mathsf{LPWW},\lambda} = \{\mathscr{L}^{\mathsf{par}}_{\mathsf{LPWW},\lambda}\}_{\mathsf{par}\in\Lambda_\lambda}$ such that $\mathscr{L}^{\mathsf{par}}_{\mathsf{LPWW},\lambda}$ is the following parametrized language:

$$\mathscr{L}^{\mathsf{par}}_{\mathsf{LPWW},\lambda} := \left\{ (g^s, g^u) \in \mathbb{G}^2 \mid \exists \mathbf{y} \in \mathbb{Z}_p^d \text{ s.t. } g^{\mathbf{y}^\top \mathbf{v}} = g^s \ \wedge \ g^{\mathbf{y}^\top \mathbf{w}} = g^u \right\}.$$

Let $\mathsf{Col}(\mathbb{G}^d) \subset \Lambda_\lambda$ denote the set of elements of the form $(g^{\mathbf{v}}, g^{\alpha\cdot\mathbf{v}})$ for some $\mathbf{v} \neq \mathbf{0}$ and $\alpha \in \mathbb{Z}_p^*$, that is, the exponents form colinear vectors over $(\mathbb{Z}_p)^d$. Observe that for any $\mathsf{par} \in \mathsf{Col}(\mathbb{G}^d)$, $\mathscr{L}^{\mathsf{par}}_{\mathsf{LPWW},\lambda}$ is a non-trivial Diffie-Hellman-style language (hence, $\mathscr{L}^{\mathsf{par}}_{\mathsf{LPWW}}$ is a sparse subset of Λ_λ); however, for any $\mathsf{par} \in \Lambda_\lambda \backslash \mathsf{Col}(\mathbb{G}^d)$, $\mathscr{L}^{\mathsf{par}}_{\mathsf{LPWW},\lambda}$ is actually equal to $\mathbb{G}^2$ (hence, $\mathscr{L}^{\mathsf{par}}_{\mathsf{LPWW},\lambda}$ is a trivial language). Below, we may omit the security parameter and use the shorthand $\mathscr{L}_{\mathsf{LPWW}} = \{\mathscr{L}^{\mathsf{par}}_{\mathsf{LPWW}}\}_{\mathsf{par}\in\Lambda}$ when the meaning is clear.

4.2 IHBG-Friendly Statistical ZAPs for the LPWW Language $\mathscr{L}_{\mathsf{LPWW}}$

Looking ahead, our construction of IHBG in Section 5 will rely at its core on an adaptively secure statistical ZAP for the family of parametrized languages $\mathscr{L}_{\mathsf{LPWW}} = \{\mathscr{L}^{\mathsf{par}}_{\mathsf{LPWW}}\}_{\mathsf{par}\in\Lambda}$. More precisely, the statistical ZAP which we will use in our construction satisfies a variant of the standard notion of adaptive computational soundness (which we defined for a single language in Section 2): we require adaptive computational soundness to hold with respect to parameters par *sampled uniformly from* $\mathsf{Col}(\mathbb{G}^d) \subset \Lambda$ (recall that $\mathsf{Col}(\mathbb{G}^d)$ is the subset of parameters such that $\mathscr{L}^{\mathsf{par}}_{\mathsf{LPWW}}$ is nontrivial). In contrast, adaptive statistical witness indistinguishability must hold *even for adversarially chosen parameters* $\mathsf{par} \in \Lambda$ (hence, in a sense, WI is doubly-adaptive: with respect to the statement, and with respect to the language parameters). We call a statistical ZAP with these properties an IHBG-*friendly* statistical ZAP for $\mathscr{L}_{\mathsf{LPWW}}$. We provide a formal definition below.

Definition. We formally introduce the notion of IHBG-friendly statistical ZAP for the family of parametrized languages $\mathscr{L}_{\mathsf{LPWW}}$.

Definition 22 (IHBG-Friendly Statistical ZAP for $\mathscr{L}_{\mathsf{LPWW}}$). *Let $\Lambda_\lambda = (\mathbb{G}^d \backslash \{\mathbf{1}\})^2$ be the parameter space for any $\lambda \in \mathbb{N}$ and consider the family of parameterized* NP *languages $\mathscr{L}_{\mathsf{LPWW}} = \{\mathscr{L}_{\mathsf{LPWW},\lambda}\}_\lambda = \{\{\mathscr{L}^{\mathsf{par}}_{\mathsf{LPWW},\lambda}\}_{\mathsf{par}\in\Lambda_\lambda}\}_\lambda$, with associated witness relation $\mathcal{R}_{\mathsf{LPWW}} = \{\mathcal{R}_{\mathsf{LPWW},\lambda}\}_\lambda = \{\{\mathcal{R}^{\mathsf{par}}_{\mathsf{LPWW},\lambda}\}_{\mathsf{par}\in\Lambda_\lambda}\}_\lambda$. Then, an* IHBG-*friendly ZAP system Π_{ZAP} for $\mathscr{L}_{\mathsf{LPWW}}$ with with public-coin length $\ell(\lambda)$ is a tuple of PPT algorithms* $(\mathsf{Prove}, \mathsf{Verify})$ *defined as follows.*

$\mathsf{Prove}(\mathsf{par}, r, x, w) \to \pi$: *The proving algorithm is given the parameters $\mathsf{par} \in \Lambda_\lambda$, the public-coin $r \in \{0,1\}^\ell$, a statement x, and a witness w, and outputs a proof π.*

$\mathsf{Verify}(\mathsf{par}, r, x, \pi) \to \top$ **or** $\bot$: *The verification algorithm is given the parameters $\mathsf{par} \in \Lambda_\lambda$, the public-coin $r \in \{0,1\}^\ell$, a statement x, and a proof π, and outputs $\top$ for acceptance or $\bot$ for rejection.*

We additionally require the following properties to hold.

Correctness: For any $\lambda \in \mathbb{N}$, $r \in \{0,1\}^\ell$, $\mathsf{par} \in \Lambda_\lambda$, and $(x, w) \in \mathcal{R}^{\mathsf{par}}_{\mathsf{LPWW},\lambda}$, we have

$$\Pr[\mathsf{Verify}(\mathsf{par}, r, x, \mathsf{Prove}(\mathsf{par}, r, x, w)) = \top] = 1.$$

(Adaptive) Computational $\varepsilon_{\mathsf{sound}}$-Soundness w.r.t. Colinear Parameters: For any $\lambda \in \mathbb{N}$ and PPT adversary $\mathcal{A}$, we have

$$\Pr\left[\mathsf{par} \leftarrow_r \mathsf{Col}(\mathbb{G}^d), r \leftarrow \{0,1\}^\ell, (x,\pi) \leftarrow_r \mathcal{A}(\mathsf{par}, r) : \begin{array}{l} x \notin \mathscr{L}^{\mathsf{par}}_{\mathsf{LPWW},\lambda} \ \wedge \\ \mathsf{Verify}(\mathsf{par}, r, x, \pi) = \top \end{array}\right] \leq \epsilon_{\mathsf{sound}}.$$

[11] To be precise, $g \in \mathbb{G}$ will be sampled for each security parameter λ and the family of parameterized language $\mathscr{L}_{\mathsf{LPWW},\lambda}$ is defined with respect to such generator g. For better readability, we may make the random sampling of g implicit when the context is clear.

(Doubly-Adaptive) Statistical Witness Indistinguishability: For any $\lambda \in \mathbb{N}$ and unbounded adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, we have

$$\left| \Pr\left[\begin{array}{l} (r, \mathsf{par}, x, w_0, w_1, \mathsf{st}) \leftarrow_r \mathcal{A}_0(1^\lambda) \\ \pi \leftarrow_r \mathsf{Prove}(\mathsf{par}, r, x, w_0) \end{array} : \begin{array}{r} \mathsf{par} \in \Lambda_\lambda \ \wedge \ \mathcal{A}_1(\mathsf{st}, \pi) = 1 \\ \wedge \ (x, w_0) \in \mathcal{R}^{\mathsf{par}}_{\mathsf{LPWW},\lambda} \\ \wedge \ (x, w_1) \in \mathcal{R}^{\mathsf{par}}_{\mathsf{LPWW},\lambda} \end{array} \right] \right|$$
$$- \left| \Pr\left[\begin{array}{l} (r, \mathsf{par}, x, w_0, w_1, \mathsf{st}) \leftarrow_r \mathcal{A}_0(1^\lambda) \\ \pi \leftarrow_r \mathsf{Prove}(\mathsf{par}, r, x, w_1) \end{array} : \begin{array}{r} \mathsf{par} \in \Lambda_\lambda \ \wedge \ \mathcal{A}_1(\mathsf{st}, \pi) = 1 \\ \wedge \ (x, w_0) \in \mathcal{R}^{\mathsf{par}}_{\mathsf{LPWW},\lambda} \\ \wedge \ (x, w_1) \in \mathcal{R}^{\mathsf{par}}_{\mathsf{LPWW},\lambda} \end{array} \right] \right| \leq \mathsf{negl}(\lambda).$$

Building IHBG-Friendly Statistical ZAPs for $\mathscr{L}_{\mathsf{LPWW}}$. In Section 6, we will provide two constructions of an IHBG-friendly statistical ZAPs for $\mathscr{L}_{\mathsf{LPWW}}$, one in pairing groups (Theorem 40), and one in pairing-free groups (Theorem 46). Both constructions are obtained by compiling the Σ-protocol for $\mathscr{L}_{\mathsf{LPWW}}$ described in Section 4.4 into an IHBG-friendly statistical ZAP for $\mathscr{L}_{\mathsf{LPWW}}$. Below, we give an overview of the main lemmas regarding our two constructions whose proofs are provided in Section 6.

Pairing-Based Construction. The pairing-based construction builds upon the Couteau-Hartmann compiler from [8], which relies on the hardness of the kernel Diffie-Hellman assumption in a group $\mathbb{G}_2$ (more generally, it can be based on the kernel k-Lin assumption in $\mathbb{G}_2$ for any k), a standard search assumption (which is implied in particular by DDH in $\mathbb{G}_2$) introduced in [38] and used in several works on pairing-based NIZKs, e.g. [33].

Lemma 23. *Let $(\mathbb{G}_1, \mathbb{G}_2)$ be bilinear-map groups equipped with an asymmetric pairing (implicitly parameterized by the security parameter λ). There exists an IHBG-friendly adaptive statistical ZAP for the family of parametrized languages $\mathscr{L}_{\mathsf{LPWW}}$ over $\mathbb{G}_1$ which satisfies adaptive computational $\varepsilon_{\mathsf{sound}}$-soundness w.r.t. colinear parameters, and doubly-adaptive statistical witness indistinguishability, assuming the explicit $\varepsilon_{\mathsf{sound}}$-hardness of the kernel Diffie-Hellman assumption in $\mathbb{G}_2$.*

Pairing-Free Construction. The pairing-free construction builds upon the compiler of [10]. The work of [10] build a correlation intractable hash function under the $2^{-3\lambda/4}$-OW-KDM security of ElGamal, which suffices to compile the above Σ-protocol into a statistical ZAP. We refine their approach and achieve a similar result under a weaker assumption, by managing to reduce the constant $3/4$ to $1/2$, that is, rely on the $2^{-\lambda/2}$-OW-KDM security of ElGamal. We note that the best known attack against this falsifiable search assumption succeeds with probability $\mathsf{poly}(\lambda)/2^\lambda$.

Lemma 24. *Let $\mathbb{G}$ be a group of order p such that $\lambda \approx 2\lceil \log p \rceil^2$. There exists an IHBG-friendly adaptive statistical ZAP for the family of parametrized languages $\mathscr{L}_{\mathsf{LPWW}}$ over $\mathbb{G}$ which satisfies adaptive computational $\varepsilon_{\mathsf{sound}}$-soundness w.r.t. colinear parameters for any $\varepsilon_{\mathsf{sound}} = 2^{-o(\lceil \log p \rceil^2)}$, and doubly-adaptive statistical witness indistinguishability, assuming the $2^{-\lambda/2}$-OW-KDM hardness of ElGamal over another group $\tilde{\mathbb{G}}$ of size $|\tilde{\mathbb{G}}| \approx 2^\lambda$.*

4.3 Σ-protocols for Parameterized Families of Languages

To construct our IHBG-friendly statistical ZAPs for $\mathscr{L}_{\mathsf{LPWW}}$, we rely on a Σ-protocol for the family of parameterized language $\mathscr{L}_{\mathsf{LPWW}}$. Here, we recall the definition of Σ-protocols, adapted to a family of parametrized languages, rather than just one single language. A Σ-protocol is a three-move interactive proof between a prover P and a verifier V for a family of parameterized languages $\mathscr{L} = \{\mathscr{L}_\lambda\}_\lambda = \{\{\mathscr{L}^{\mathsf{par}}_\lambda\}_{\mathsf{par} \in \Lambda_\lambda}\}_\lambda$ with associated witness relation $\mathcal{R} = \{\mathcal{R}_\lambda\}_\lambda = \{\{\mathcal{R}^{\mathsf{par}}_\lambda\}_{\mathsf{par} \in \Lambda_\lambda}\}_\lambda$. The prover sends an initial message $\alpha \in I_\lambda$ for some commitment space I_λ, the verifier responds with a random $\beta \leftarrow_r S_\lambda$ for some challenge space S_λ, and the prover concludes with a message γ. Lastly, the verifier outputs $\top$, if it accepts and $\bot$ otherwise. We denote $\mathsf{out}\langle \mathsf{P}(\mathsf{par}, x, w), \mathsf{V}(\mathsf{par}, x)\rangle$ as the final value output by the verifier and $\mathsf{trans}\langle \mathsf{P}(\mathsf{par}, x, w), \mathsf{V}(\mathsf{par}, x)\rangle$ as the transcript (α, β, γ). A Σ-protocol is usually defined by three properties: completeness, special honest-verifier zero-knowledge, and special soundness (see for example [35]). In this work, we will instead be interested in statistical witness indistinguishability even against malicious verifiers and adaptive soundness. The definitions follow.

Definition 25 (Completeness). *For any* $\lambda \in \mathbb{N}$, $\mathsf{par} \in \Lambda_\lambda$ *and* $(x, w) \in \mathcal{R}_\lambda^{\mathsf{par}}$, *we have* $\Pr[\mathsf{out}\langle \mathsf{P}(\mathsf{par}, x, w), \mathsf{V}(\mathsf{par}, x)\rangle = \top] = 1$.

Definition 26 (Statistical Witness Indistinguishability). *For any* $\lambda \in \mathbb{N}$, $\mathsf{par} \in \Lambda_\lambda$ *and* (x, w_0, w_1) *such that* $(x, w_0) \in \mathcal{R}_\lambda^{\mathsf{par}}$ *and* $(x, w_1) \in \mathcal{R}_\lambda^{\mathsf{par}}$, *and for any (possibly inefficient) cheating verifier* V^*, *the two distributions* $\mathsf{trans}\langle \mathsf{P}(\mathsf{par}, x, w_0), \mathsf{V}^*(\mathsf{par}, x)\rangle$ *and* $\mathsf{trans}\langle \mathsf{P}(\mathsf{par}, x, w_1), \mathsf{V}^*(\mathsf{par}, x)\rangle$ *are statistically close. We say it is perfect witness indistinguishability when the two distributions are identical.*

Definition 27 (Adaptive ρ-soundness). *For any* $\lambda \in \mathbb{N}$, $\mathsf{par} \in \Lambda_\lambda$, *any (possibly inefficient) cheating prover* P^*, *and any first flow* α, *it holds that* $\Pr[\beta \leftarrow_r S_\lambda;\ (x, \gamma) \leftarrow_r \mathsf{P}^*(\mathsf{par}, \alpha, \beta) :\ x \notin \mathscr{L}_\lambda^{\mathsf{par}} \wedge \mathsf{V}(\mathsf{par}, x, \alpha, \beta, \gamma) = \top] \leq \rho(\lambda)$.

4.4 A Σ-Protocol for the LPWW Language $\mathscr{L}_{\mathsf{LPWW}}$

Fix some parameters $\mathsf{par} = (g^{\mathbf{v}}, g^{\mathbf{w}}) \in \Lambda = (\mathbb{G}^d \backslash \{\mathbf{1}\})^2$ (implicitly parameterized by the security parameter λ). To match with the notations which we will use later when building an IHBG, we denote the dimension d in $\mathscr{L}_{\mathsf{LPWW}}^{\mathsf{par}}$ by $m+1$. We consider a statement $(\hat{X}, \hat{Y}) := (g^{\hat{x}}, g^{\hat{y}}) \in \mathscr{L}_{\mathsf{LPWW}}^{\mathsf{par}}$ and let $\mathbf{y} \in \mathbb{Z}_p^{m+1}$ be the prover witness (i.e., $\mathbf{y}$ is any vector over $\mathbb{Z}_p^{m+1}$ such that $\mathbf{y}^\top \mathbf{v} = \hat{x}$ and $\mathbf{y}^\top \mathbf{w} = \hat{y}$). Let $n \in \mathbb{N}$ be any positive integer. Then, a Σ-protocol for $\mathscr{L}_{\mathsf{LPWW}} = \{\mathscr{L}_{\mathsf{LPWW}}^{\mathsf{par}}\}_{\mathsf{par} \in \Lambda}$ is provided in Figure 1. Correctness can be checked by routine calculation. Below, we prove statistical witness indistinguishability and adaptive soundness.

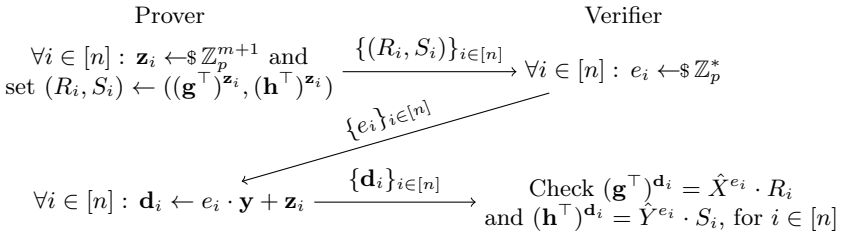

Fig. 1. Σ-protocol with statement $(\hat{X}, \hat{Y}) \in \mathscr{L}_{\mathsf{LPWW}}^{\mathsf{par}}$ where $\mathsf{par} := (\mathbf{g}, \mathbf{h}) = (g^{\mathbf{v}}, g^{\mathbf{w}})$.

Lemma 28 (Perfect Witness Indistinguishability). *The* IHBG*-friendly* Σ*-protocol for the family of parametrized languages* $\mathscr{L}_{\mathsf{LPWW}} = \{\mathscr{L}_{\mathsf{LPWW}}^{\mathsf{par}}\}_{\mathsf{par} \in \Lambda}$ *in Figure 1 satisfies perfect witness indistinguishability.*

Proof. Fix any $\mathsf{par} = (g^{\mathbf{v}}, g^{\mathbf{w}}) \in \Lambda$ and statement $(\hat{X}, \hat{Y}) = (g^{\hat{x}}, g^{\hat{y}}) \in \mathscr{L}_{\mathsf{LPWW}}^{\mathsf{par}}$ such that the set of witnesses $\mathcal{W} := \{\mathbf{y} \in \mathbb{Z}_p^{m+1} \mid \mathbf{y}^\top \mathbf{v} = \hat{x} \wedge \mathbf{y}^\top \mathbf{w} = \hat{y}\}$ is non-trivial, i.e., there exists at least two witnesses in $\mathcal{W}$. Below, we only consider the case $n = 1$ since witness indistinguishability is closed under parallel repetition. First, notice that the distribution of the first message (R, S) is independent of the witnesses. This in particular implies that the output e of the (possibly inefficient) cheating verifier V^* is also independent of the witnesses as well. Therefore, it suffices to show that the distribution of the third message $\mathbf{d}$ is identical for any witness $\mathbf{y} \in \mathcal{W}$ conditioned on an arbitral fixed choice of $(R, S) = (g^r, g^s)$ and e. Let $D_{\mathbf{z}}$ be the distribution of sampling a uniformly random $\mathbf{z} \in \mathbb{Z}_p^{m+1}$ conditioned on $\mathbf{v}^\top \mathbf{z} = r$ and $\mathbf{w}^\top \mathbf{z} = s$. Then, for any witness $\mathbf{y} \in \mathcal{W}$, the third message $\mathbf{d}$ follows the distribution $D_{\mathbf{d}}^{\mathbf{y}}$ where we first sample $\mathbf{z} \leftarrow_r D_{\mathbf{z}}$ and output $\mathbf{d} \leftarrow e \cdot \mathbf{y} + \mathbf{z}$. Observe that by the definition of $D_{\mathbf{z}}$, the probability of any $\mathbf{d}$ being sampled form $D_{\mathbf{d}}^{\mathbf{y}}$ and $D_{\mathbf{d}}^{\mathbf{y}'}$ for any two witnesses $\mathbf{y}, \mathbf{y}' \in \mathcal{W}$ is identical. In particular, we have $D_{\mathbf{d}}^{\mathbf{y}} = D_{\mathbf{d}}^{\mathbf{y}'}$ for any $\mathbf{y}, \mathbf{y}' \in \mathcal{W}$. Combining all the arguments together, the statement follows.

Lemma 29 (Adaptive Soundness). *The* IHBG*-friendly* Σ*-protocol for the family of parametrized languages* $\mathscr{L}_{\mathsf{LPWW}} = \{\mathscr{L}_{\mathsf{LPWW}}^{\mathsf{par}}\}_{\mathsf{par} \in \Lambda}$ *in Figure 1 satisfies adaptive* $(\frac{1}{p-1})^{n-1}$*-soundness.*

Proof. In case $\mathsf{par} \in \Lambda \backslash \mathsf{Col}(\mathbb{G}^d)$, we have $\mathscr{L}_{\mathsf{LPWW}}^{\mathsf{par}} = \mathbb{G}^2$ so there exists no statement $(\hat{X}, \hat{Y}) \notin \mathscr{L}_{\mathsf{LPWW}}^{\mathsf{par}}$ (see Section 4.1). Therefore, we only need to consider the case $\mathsf{par} \in \mathsf{Col}(\mathbb{G}^d)$. For any such $\mathsf{par} =$

$(g^{\mathbf{v}}, g^{\mathbf{w}})$, we have $\mathbf{v} \neq \mathbf{0}$ and $\mathbf{w} = \alpha \cdot \mathbf{v}$ for some $\alpha \in \mathbb{Z}_p^*$. Now, fix any first flow $\{(R_i, S_i)\}_{i\in[n]}$ and let r_i and s_i be their exponents, respectively, i.e., $R_i = g^{r_i}$ and $S_i = g^{s_i}$ and assume the adversary outputs $(\hat{X}, \hat{Y}) = (g^{\hat{x}}, g^{\hat{y}}) \notin \mathscr{L}^{\mathsf{par}}_{\mathsf{LPWW}}$. In order for the verifier to accept, the following must hold: $e_i \cdot \hat{y} + s_i = e_i \cdot \hat{x} \cdot \alpha + r_i \cdot \alpha$, for $i \in [n]$, which implies that:

$$\hat{y} - \alpha \cdot \hat{x} = (\alpha \cdot r_i - s_i) \cdot e_i^{-1}, \text{ for } i \in [n]$$

We distinguish the following two cases:

1. If $\alpha \cdot r_i = s_i$ for some $i \in [n]$, then $\hat{y} = \alpha \cdot \hat{x}$. This implies $(g^{\hat{x}}, g^{\hat{y}}) \in \mathscr{L}^{\mathsf{par}}_{\mathsf{LPWW}}$.
2. If $\alpha \cdot r_i \neq s_i$ for all $i \in [n]$, then $\frac{\alpha \cdot r_1 - s_1}{\alpha \cdot r_i - s_i} \cdot e_i = e_1$ for $i \in [n]$, which happens with probability at most $(\frac{1}{p-1})^{n-1}$ over the randomness of $(e_i)_{i\in[n]} \leftarrow_r (\mathbb{Z}_p^*)^n$.

We can ignore the first case since it does not lead to a valid cheating prover for adaptive soundness. In the second case, even a computationally unbounded cheating prover has at most $(\frac{1}{p-1})^{n-1}$ possibility of breaking adaptive soundness. Hence the statement follows.

5 Interactive Hidden-Bits Generating Protocols from the Explicit Hardness of DDH and an IHBG-Friendly Statistical ZAPs for $\mathscr{L}_{\mathsf{LPWW}}$

In this section, we construct an IHBG protocol based on explicit μ-hardness of the DDH assumption (over a pairing-free group, for a negligible function μ arbitrarily close to an inverse polynomial function) and an IHBG-friendly statistical ZAP for the language $\mathscr{L}_{\mathsf{LPWW}}$, defined in Section 4, which is naturally induced from the (non-interactive) hidden-bits generator of Libert et al. [34].

5.1 Constructing the IHBG Protocol

Building Block. Our construction is parametrized by λ and $\mu(\lambda)$, and relies on the following building blocks:

- $\mathcal{H} = \{\mathcal{H}_\lambda\}_\lambda = \{\{H : \mathbb{G} \mapsto \{0,1\}\}_H\}_\lambda$ is a family of universal hash functions with description size of at most $O(\log_2 p)$ bits, where $\mathbb{G}$ and p are implicitly parameterized by the security parameter.
- $\Pi_{\mathsf{ZAP}} = (\mathsf{ZAP.Prove}, \mathsf{ZAP.Verify})$ is an IHBG-friendly ZAP for the parametrized family of languages $\mathscr{L}_{\mathsf{LPWW}} = \{\mathscr{L}_{\mathsf{LPWW},\lambda}\}_\lambda = \{\{\mathscr{L}^{\mathsf{par}}_{\mathsf{LPWW},\lambda}\}_{\mathsf{par}\in\Lambda_\lambda}\}_\lambda$ with public-coin length $\ell'(\lambda)$, satisfying adaptive computational $\epsilon_{\mathsf{sound}}$-soundness w.r.t. colinear parameters for $\epsilon_{\mathsf{sound}} = \frac{\mu(\lambda)}{m(\lambda)}$ and doubly-adaptive statistical witness indistinguishability. Here, we set the vector length parameter $d(\lambda)$ in $\mathscr{L}_{\mathsf{LPWW},\lambda}$ to $m(\lambda)+1$, where $m(\lambda)$ is the polynomial output bit length of the IHBG protocol defined below.

Construction. The construction of an IHBG protocol denoted as Π_{IHBG} is described as follows. The commitment length is at most $s(\lambda) = \lceil \log_2 p \rceil + O(\log_2 p)$ where $(\mathbb{G}, p) \leftarrow \mathsf{DHGen}(1^\lambda)$ (note that DHGen guarantees in particular $p > \lambda^{\omega(1)}$, which is needed to use the uniformity property of H). The output bit length $m(\lambda)$ is an arbitrary large enough fixed polynomial $\mathsf{poly}(\lambda)$, and the public-coin length $\ell(\lambda)$ is $m \cdot \ell' + (m+2) \cdot \lceil \log_2 p \rceil$. We rely on one more parameter $\nu(\lambda)$ and require the parameters to satisfy the following conditions:

- In order to prove statistical hiding, $m(\lambda) \cdot \mu(\lambda)$ must be negligible; this holds by setting $\mu(\lambda)$ to be a negligible function.
- For technical reasons in the hybrid games, we need a negligible gap between ν and μ; that is, $\nu(\lambda)$ is a negligible function satisfying $\mu(\lambda) = \nu(\lambda) \cdot \mathsf{negl}(\lambda)$.
- We also need $1/\mu(\lambda)$ (and hence $1/\nu(\lambda)$) to be small compared to p (otherwise, assuming explicit μ-hardness of DDH over $\mathbb{G}$ does not make sense: a polynomial time attack with $O(1/p)$ advantage against DDH trivially exists). In particular, $\mu(\lambda)$ can be set as an arbitrary close to an inverse polynomial, i.e., $\lambda^{-\omega(1)}$. Here, since $1/\nu(\lambda)$ is small compared to p, any element $z \in [1/\nu(\lambda)]$ can be seen as an element of $\mathbb{Z}_p$.

We proceed with the description of the scheme. In the following we may omit the dependency on λ for better readability when the context is clear.

$\mathsf{GenBits}(1^\lambda, m, r)$: On input the security parameter 1^λ, bit length m, and a public-coin $r \in \{0,1\}^\ell$, parse $((r_{\mathsf{ZAP},i})_{i\in[m]}, g, g^{\mathbf{M}}) \leftarrow r$, where $g \in \mathbb{G}$ and $\mathbf{M} := (\mathbf{v}|\mathbf{w}_1|\dots|\mathbf{w}_m) \in \mathbb{Z}_p^{(m+1)\times(m+1)}$.[12] Then sample $z \leftarrow_r [1/\nu]$, and compute $g^{\mathbf{M}-z\cdot\mathbf{I}_{m+1}}$, where we denote $\mathbf{M}' := \mathbf{M} - z\cdot\mathbf{I}_{m+1} = (\mathbf{v}'|\mathbf{w}'_1|\cdots|\mathbf{w}'_m) \in \mathbb{Z}_p^{(m+1)\times(m+1)}$. Further sample a random hash function $H \leftarrow_r \mathcal{H}$ and a uniformly random seed $\mathbf{y} \leftarrow_r \mathbb{Z}_p^{m+1}$, and compute a commitment $g^s \leftarrow g^{\mathbf{y}^\top\mathbf{v}'}$, openings $g^{u_i} \leftarrow g^{\mathbf{y}^\top\mathbf{w}'_i}$, and the hidden bits $\rho_i \leftarrow H(g^{u_i})$ for all $i \in [m]$. For each $i \in [m]$, set the language parameter $\mathsf{par}_i := (g^{\mathbf{v}'}, g^{\mathbf{w}'_i})$, statement $x_i := (g^s, g^{u_i})$, and witness $w := \mathbf{y}$ for membership to the parametrized language $\mathscr{L}^{\mathsf{par}_i}_{\mathsf{LPWW}}$, and compute $\pi_{\mathsf{ZAP},i} \leftarrow_r \mathsf{ZAP.Prove}(\mathsf{par}_i, r_{\mathsf{ZAP},i}, x_i, w)$ and set $\pi_i = (g^{u_i}, \pi_{\mathsf{ZAP},i})$. Finally, output the commitment $\sigma := (H, g^s, z) \in \mathcal{H}\times\mathbb{G}\times[1/\nu]$, string $\rho := (\rho_i)_{i\in[m]} \in \{0,1\}^m$ and the set of proofs $\{\pi_i\}_{i\in[m]}$.

$\mathsf{VerifyBit}(r, \sigma, i, \rho_i, \pi_i)$: Parse $((r_{\mathsf{ZAP},i})_{i\in[m]}, g, g^{\mathbf{M}}) \leftarrow r$, $(H, g^s, z) \leftarrow \sigma$, $(g^{u_i}, \pi_{\mathsf{ZAP},i}) \leftarrow \pi_i$, and compute $g^{\mathbf{M}'} \leftarrow g^{\mathbf{M}-z\mathbf{I}_{m+1}}$. Then, set the language parameter as $\mathsf{par}_i := (g^{\mathbf{v}'}, g^{\mathbf{w}'_i})$ and the statement as $x_i := (g^s, g^{u_i})$. Check $\rho_i = H(g^{u_i})$ and $\mathsf{ZAP.Verify}(\mathsf{par}_i, r_{\mathsf{ZAP},i}, x_i, \pi_{\mathsf{ZAP},i}) = \top$. Output $\top$ if both check passes and otherwise output $\bot$.

Succinctness The length of the commitment $\sigma = (H, g^s, z)$ only depends on the security parameter, and in particular, independent of m. This is because g^s requires $\lceil\log_2 p\rceil$ bits, z requires $\lceil\log_2(1/\nu(\lambda))\rceil \le \lceil\log_2 p\rceil$ and the description of the universal hash function H requires at most $O(\log_2 p)$ bits.

5.2 Security

Correctness of our IHBG protocol can be verified by a routine check. Below, we show our IHBG protocol satisfies extractability and statistical hiding in the following Theorems 30 and 32.

Theorem 30 (Extractability). *Consider $\mu(\lambda)$ an efficiently computable function, $\varepsilon_{\mathsf{sound}} = \frac{\nu(\lambda)}{m(\lambda)}$, and a negligible function $\nu(\lambda)$ such that $\mu(\lambda) = \nu(\lambda)\cdot\mathsf{negl}(\lambda)$. If the IHBG-friendly ZAP for $\mathscr{L}_{\mathsf{LPWW}}$ is adaptively computational $\varepsilon_{\mathsf{sound}}$-sound w.r.t. colinear parameters and the DDH assumption is μ-explicitly hard, then IHBG satisfies ν-extractability.*

Proof. We show that our IHBG satisfies public-coin indistinguishability and μ-successful extraction.

Public-Coin Indistinguishability. We first describe the PPT algorithm SimCoin. SimCoin starts by sampling $g \leftarrow_r \mathbb{G}$, $\mathbf{v}' \leftarrow_r \mathbb{Z}_p^{m+1}\backslash\{\mathbf{0}\}$ and $\alpha_i \leftarrow_r \mathbb{Z}_p^*$, for all $i \in [m]$, and sets $\mathbf{w}'_i = \alpha_i\mathbf{v}'$ and $\mathbf{M}'' := (\mathbf{v}'|\mathbf{w}'_1|\dots|\mathbf{w}'_m) \in \mathbb{Z}_p^{(m+1)\times(m+1)}$. It then samples $\tilde{z} \leftarrow_r [1/\nu]$, $r_{\mathsf{ZAP},i} \leftarrow_r \{0,1\}^{\ell'}$ for all $i \in [m]$. It now computes $\mathbf{M} := \mathbf{M}'' + \tilde{z}\cdot\mathbf{I}_{m+1}$, interprets $((r_{\mathsf{ZAP},i})_{i\in[m]}, g, g^{\mathbf{M}})$ as a binary string $\widetilde{r} \in \{0,1\}^\ell$. Finally, it outputs the simulated public-coin $\widetilde{r}$ and sets the trapdoor $\tau := ((\alpha_i)_{i\in[m]}, \tilde{z})$. Following textbook arguments, the following holds due to the polynomial hardness of DDH:

$$\left|\Pr[r \leftarrow_r \{0,1\}^\ell : \mathcal{A}(m,r) = 1]\right| - \left|\Pr[(\widetilde{r},\tau) \leftarrow_r \mathsf{SimCoin}(1^\lambda, m) : \mathcal{A}(m,\widetilde{r}) = 1]\right| \le \mathsf{negl}(\lambda).$$

μ-Successful Extraction. We first describe the efficient, deterministic $\mathsf{Open}(\widetilde{r}, \sigma, \tau)$ algorithm:

- Parse $(H, g^s, z) \leftarrow \sigma$ and $((\alpha_i)_{i\in[m]}, \tilde{z}) \leftarrow \tau$.
- If $\tilde{z} \neq z$, then return $\bot$.
- Otherwise, for each bit $i \in [m]$, compute $\rho_i := H(g^{s\cdot\alpha_i})$.
- Return $\rho = (\rho_i)_{i\in[m]} \in \{0,1\}^m$.

We proceed by a hybrid argument between PPT adversary $\mathcal{A}$, PPT distinguisher D, and a challenger. We denote E_i as the event that the challenger outputs 1 in Game_i. Looking ahead, the probability of E occurring in the first and last games correspond to the probability of the left-hand and right-hand side of the ν-successful extraction, respectively.

Game_0 : In this game, the challenger first runs $(\widetilde{r}, \tau) \leftarrow \mathsf{SimCoin}(1^\lambda, m)$ and invokes the adversary $\mathcal{A}$ on input $(m, \widetilde{r})$. Once $\mathcal{A}$ replies with $(\sigma, S, \rho^*_S, \{\pi_i\}_{i\in S}, \mathsf{st})$, the challenger computes $\rho \leftarrow \mathsf{Open}(\widetilde{r}, \sigma, \tau)$. The challenger outputs 1 if the following holds: $D(\mathsf{st}) = 1$, $\rho \in \{0,1\}^m$, and

[12] Note the algorithm only has knowledge of the encodings $g^{\mathbf{M}}$, and does not know the discrete logarithms $\mathbf{M}$ themselves.

$\mathsf{VerifyBit}(\widetilde{r}, \sigma, i, 1-\rho_i, \pi_i) = \perp$ for all $i \in S$. The probability the challenger outputs 1 corresponds to the left-hand side of the ν-successful extraction condition in Definition 14. Let us denote

$$\Pr[\mathsf{E}_0] := \varepsilon.$$

Below, we give the full description of the challenger, which receives as input the security parameter λ (in unary) and polynomial m, and proceeds as follows:

1. $g \leftarrow_r \mathbb{G}$
2. $\mathbf{v}' \leftarrow_r \mathbb{Z}_p^{m+1} \backslash \{\mathbf{0}\}$
3. $\alpha_i \leftarrow_r \mathbb{Z}_p^*$, for all $i \in [m]$.
4. $\mathbf{w}'_i \leftarrow \alpha_i \mathbf{v}'$, for all $i \in [m]$.
5. $\mathbf{M}'' \leftarrow (\mathbf{v}' | \mathbf{w}'_1 | \ldots | \mathbf{w}'_m) \in \mathbb{Z}_p^{(m+1)\times(m+1)}$.
6. $\tilde{z} \leftarrow_r [1/\nu]$
7. $\mathbf{M} \leftarrow \mathbf{M}'' + \tilde{z} \cdot \mathbf{I}_{m+1}$.
8. $r_{\mathsf{ZAP},i} \leftarrow_r \{0,1\}^{\ell'}$, for all $i \in [m]$
9. Set the simulated public-coin as $\widetilde{r} := ((r_{\mathsf{ZAP},i})_{i\in[m]}, g, g^{\mathbf{M}}) \in \{0,1\}^{\ell}$.
10. Set the extraction trapdoor as $\tau := ((\alpha_i)_{i\in[m]}, \tilde{z})$.
11. Send $(m, \widetilde{r})$ to $\mathcal{A}$, which replies with $(\sigma, S, \rho_S^*, \{\pi_i\}_{i\in S}, \mathsf{st})$.
12. Run $\rho \leftarrow \mathsf{Open}(\widetilde{r}, \sigma, \tau)$ and if $\rho = \perp$, return 0.
13. Send st to D. If D replies with 0, return 0.
14. Check $\mathsf{VerifyBit}(\widetilde{r}, \sigma, i, 1-\rho_i, \pi_i) = \perp$, for all $i \in S$. If not, return 0.
15. Return 1.

Game_1 : This game is defined identically to Game_0, except that the challenger now performs an additional check whether $\tilde{z}$ picked by the $\mathsf{SimCoin}$ algorithm is the same as the z it received from the adversary $\mathcal{A}$. Since Open already makes this check, this is just a conceptual change and we have

$$\Pr[\mathsf{E}_1] = \Pr[\mathsf{E}_0].$$

Below, we give the full description of the challenger, where the red underline indicates the difference between the previous game:

1. $g \leftarrow_r \mathbb{G}$
2. $\mathbf{v}' \leftarrow_r \mathbb{Z}_p^{m+1} \backslash \{\mathbf{0}\}$
3. $\alpha_i \leftarrow_r \mathbb{Z}_p^*$, for all $i \in [m]$.
4. $\mathbf{w}'_i \leftarrow \alpha_i \mathbf{v}'$, for all $i \in [m]$.
5. $\mathbf{M}'' \leftarrow (\mathbf{v}' | \mathbf{w}'_1 | \ldots | \mathbf{w}'_m) \in \mathbb{Z}_p^{(m+1)\times(m+1)}$.
6. $\tilde{z} \leftarrow_r [1/\nu]$
7. $\mathbf{M} \leftarrow \mathbf{M}'' + \tilde{z} \cdot \mathbf{I}_{m+1}$.
8. $r_{\mathsf{ZAP},i} \leftarrow_r \{0,1\}^{\ell'}$, for all $i \in [m]$
9. Set the simulated public-coin as $\widetilde{r} := ((r_{\mathsf{ZAP},i})_{i\in[m]}, g, g^{\mathbf{M}}) \in \{0,1\}^{\ell}$.
10. Set the extraction trapdoor as $\tau := ((\alpha_i)_{i\in[m]}, \tilde{z})$.
11. Send $(m, \widetilde{r})$ to $\mathcal{A}$, which replies with $(\sigma, S, \rho_S^*, \{\pi_i\}_{i\in S}, \mathsf{st})$.
12. Parse $(H, g^s, z) \leftarrow \sigma$.
13. If $\tilde{z} \neq z$, return 0.
14. Run $\rho \leftarrow \mathsf{Open}(\widetilde{r}, \sigma, \tau)$ and if $\rho = \perp$, return 0.
15. Send st to D. If D replies with 0, return 0.
16. Check $\mathsf{VerifyBit}(\widetilde{r}, \sigma, i, 1-\rho_i, \pi_i) = \perp$, for all $i \in S$. If not, return 0.
17. Return 1.

Game_2 : This game is defined identically to Game_1, except that the challenger does not verify whether for all $i \in S$, $\mathsf{VerifyBit}(\widetilde{r}, \sigma, i, 1-\rho_i, \pi_i) = \perp$. It still checks however, if $\rho \in \{0,1\}^m$ (which is implied by the check $\tilde{z} = z$). We prove in Lemma 31 that assuming the IHBG-friendly ZAP for $\mathscr{L}_{\mathsf{LPWW}}$ is adaptively computational $\varepsilon_{\mathsf{sound}}$-sound w.r.t. colinear parameters, we have

$$|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_2]| \leq m \cdot \varepsilon_{\mathsf{sound}}.$$

So as not to interrupt the main proof, we skip the proof of Lemma 31 to later.
Below, we give the full description of the challenger:

1. $g \leftarrow_r \mathbb{G}$

2. $\mathbf{v}' \leftarrow_r \mathbb{Z}_p^{m+1} \backslash \{\mathbf{0}\}$
3. $\alpha_i \leftarrow_r \mathbb{Z}_p^*$, for all $i \in [m]$.
4. $\mathbf{w}'_i \leftarrow \alpha_i \mathbf{v}'$, for all $i \in [m]$.
5. $\mathbf{M}'' \leftarrow (\mathbf{v}'|\mathbf{w}'_1| \dots |\mathbf{w}'_m) \in \mathbb{Z}_p^{(m+1)\times(m+1)}$.
6. $\tilde{z} \leftarrow_r [1/\nu]$
7. $\mathbf{M} \leftarrow \mathbf{M}'' + \tilde{z} \cdot \mathbf{I}_{m+1}$.
8. $r_{\mathsf{ZAP},i} \leftarrow_r \{0,1\}^{\ell'}$, for all $i \in [m]$
9. Set the simulated public-coin as $\widetilde{r} := ((r_{\mathsf{ZAP},i})_{i\in[m]}, g, g^{\mathbf{M}}) \in \{0,1\}^{\ell}$.
10. Set the extraction trapdoor as $\tau := ((\alpha_i)_{i\in[m]}, \tilde{z})$.
11. Send $(m, \widetilde{r})$ to $\mathcal{A}$, which replies with $(\sigma, S, \rho_S^*, \{\pi_i\}_{i\in S}, \mathsf{st})$.
12. Parse $(H, g^s, z) \leftarrow \sigma$.
13. If $\tilde{z} \neq z$, return 0.
14. Run $\rho \leftarrow \mathsf{Open}(\widetilde{r}, \sigma, \tau)$ and if $\rho = \bot$, return 0.
15. Send st to D. If D replies with 0, return 0.
16. ~~Check $\mathsf{VerifyBit}(\widetilde{r}, \sigma, i, 1-\rho_i, \pi_i) = \bot$, for all $i \in S$. If not, return 0.~~
17. Return 1.

Game_3 : This game is defined identically to Game_2, except that the challenger no longer computes $\rho \leftarrow \mathsf{Open}(\widetilde{r}, \sigma, \tau)$ and checks $\rho = \bot$. Since in Game_2, the check $\tilde{z} \neq z$ (which is equivalent to the check $\rho = \bot$) is done prior to calling Open, and because ρ is not used at any point in Game_2, this is simply a syntactic change and the two games are equivalent and we have

$$\Pr[\mathsf{E}_2] = \Pr[\mathsf{E}_3].$$

Below, we give the full description of the challenger:

1. $g \leftarrow_r \mathbb{G}$
2. $\mathbf{v}' \leftarrow_r \mathbb{Z}_p^{m+1} \backslash \{\mathbf{0}\}$
3. $\alpha_i \leftarrow_r \mathbb{Z}_p^*$, for all $i \in [m]$.
4. $\mathbf{w}'_i \leftarrow \alpha_i \mathbf{v}'$, for all $i \in [m]$.
5. $\mathbf{M}'' \leftarrow (\mathbf{v}'|\mathbf{w}'_1| \dots |\mathbf{w}'_m) \in \mathbb{Z}_p^{(m+1)\times(m+1)}$.
6. $\tilde{z} \leftarrow_r [1/\nu]$
7. $\mathbf{M} \leftarrow \mathbf{M}'' + \tilde{z} \cdot \mathbf{I}_{m+1}$.
8. $r_{\mathsf{ZAP},i} \leftarrow_r \{0,1\}^{\ell'}$, for all $i \in [m]$
9. Set the simulated public-coin as $\widetilde{r} := ((r_{\mathsf{ZAP},i})_{i\in[m]}, g, g^{\mathbf{M}}) \in \{0,1\}^{\ell}$.
10. ~~Set the extraction trapdoor as $\tau := ((\alpha_i)_{i\in[m]}, \tilde{z})$.~~
11. Send $(m, \widetilde{r})$ to $\mathcal{A}$, which replies with $(\sigma, S, \rho_S^*, \{\pi_i\}_{i\in S}, \mathsf{st})$.
12. Parse $(H, g^s, z) \leftarrow \sigma$.
13. If $\tilde{z} \neq z$, return 0.
14. ~~Run $\rho \leftarrow \mathsf{Open}(\widetilde{r}, \sigma, \tau)$ and if $\rho = \bot$, return 0.~~
15. Send st to D. If D replies with 0, return 0.
16. Return 1.

Game_4 : This game is defined identically to Game_3, except that the challenger samples a uniformly random matrix $\mathbf{M}'' \leftarrow_r \mathbb{Z}_p^{(m+1)\times(m+1)}$ rather than running $\mathsf{SimCoin}$. As in Game_3, the challenger still samples $\tilde{z} \leftarrow_r [1/\nu]$ and checks whether $\tilde{z} = z$. Identically to the proof for public-coin indistinguishability, due to the μ-explicit hardness of DDH[13], we have

$$|\Pr[\mathsf{E}_3] - \Pr[\mathsf{E}_4]| \leq \mu(\lambda).$$

Below, we give the full description of the challenger:

1. $g \leftarrow_r \mathbb{G}$
2. $\mathbf{M}'' \leftarrow_r \mathbb{Z}_p^{(m+1)\times(m+1)}$.
3. $\tilde{z} \leftarrow_r [1/\nu]$
4. $\mathbf{M} \leftarrow \mathbf{M}'' + \tilde{z} \cdot \mathbf{I}_{m+1}$.
5. $r_{\mathsf{ZAP},i} \leftarrow_r \{0,1\}^{\ell'}$, for all $i \in [m]$

[13] Note that we need μ-explicit hardness rather than the standard polynomial hardness to provide a stricter bound between $\Pr[\mathsf{E}_3]$ and $\Pr[\mathsf{E}_4]$.

6. Set the simulated public-coin as $\widetilde{r} := ((r_{\mathsf{ZAP},i})_{i\in[m]}, g, g^{\mathbf{M}}) \in \{0,1\}^{\ell}$.
7. Send $(m, \widetilde{r})$ to $\mathcal{A}$, which replies with $(\sigma, S, \rho_S^*, \{\pi_i\}_{i\in S}, \mathsf{st})$.
8. Parse $(H, g^s, z) \leftarrow \sigma$.
9. If $\tilde{z} \neq z$, return 0.
10. Send st to D. If D replies with 0, return 0.
11. Return 1.

Game_5 : This game is defined identically to Game_4, except that the challenger directly samples a uniformly random matrix $\mathbf{M} \leftarrow_r \mathbb{Z}_p^{(m+1)\times(m+1)}$. Since the distributions of $\mathbf{M}$ are identical in both Game_4 and Game_5, we have

$$\Pr[\mathsf{E}_4] = \Pr[\mathsf{E}_5].$$

At this point, notice that $\tilde{z}$ is information theoretically hidden from $\mathcal{A}$ and D. Therefore, the probability $\tilde{z} = z$ is ν regardless of all the other randomness.
The description of the challenger is as follows:

1. $g \leftarrow_r \mathbb{G}$
2. ~~$\mathbf{M}'' \leftarrow_r \mathbb{Z}_p^{(m+1)\times(m+1)}$~~.
3. $\tilde{z} \leftarrow_r [1/\nu]$
4. $\mathbf{M} \leftarrow_r \mathbb{Z}_p^{(m+1)\times(m+1)}$.
5. $r_{\mathsf{ZAP},i} \leftarrow_r \{0,1\}^{\ell'}$, for all $i \in [m]$
6. Set the simulated public-coin as $\widetilde{r} := ((r_{\mathsf{ZAP},i})_{i\in[m]}, g, g^{\mathbf{M}}) \in \{0,1\}^{\ell}$.
7. Send $(m, \widetilde{r})$ to $\mathcal{A}$, which replies with $(\sigma, S, \rho_S^*, \{\pi_i\}_{i\in S}, \mathsf{st})$.
8. Parse $(H, g^s, z) \leftarrow \sigma$.
9. If $\tilde{z} \neq z$, return 0.
10. Send st to D. If D replies with 0, return 0.
11. Return 1.

Game_6 : This game is defined identically to Game_5, except that matrix $\mathbf{M}$ is again computed as $\mathbf{M} \leftarrow \mathbf{M}'' + \tilde{z}' \cdot \mathbf{I}_{m+1}$ *for some other* $\tilde{z}'$ sampled uniformly and independently of $\tilde{z}$. Since matrix $\mathbf{M}''$ remains uniformly random, the distributions of Game_5 and Game_6 are identical. Hence,

$$\Pr[\mathsf{E}_5] = \Pr[\mathsf{E}_6].$$

The description of the challenger is as follows:

1. $g \leftarrow_r \mathbb{G}$
2. $\mathbf{M}'' \leftarrow_r \mathbb{Z}_p^{(m+1)\times(m+1)}$.
3. $\tilde{z} \leftarrow_r [1/\nu]$
4. $\tilde{z}' \leftarrow_r [1/\nu]$
5. $\mathbf{M} \leftarrow \mathbf{M}'' + \tilde{z}' \cdot \mathbf{I}_{m+1}$.
6. $r_{\mathsf{ZAP},i} \leftarrow_r \{0,1\}^{\ell'}$, for all $i \in [m]$
7. Set the simulated public-coin as $\widetilde{r} := ((r_{\mathsf{ZAP},i})_{i\in[m]}, g, g^{\mathbf{M}}) \in \{0,1\}^{\ell}$.
8. Send $(m, \widetilde{r})$ to $\mathcal{A}$, which replies with $(\sigma, S, \rho_S^*, \{\pi_i\}_{i\in S}, \mathsf{st})$.
9. Parse $(H, g^s, z) \leftarrow \sigma$.
10. If $\tilde{z} \neq z$, return 0.
11. Send st to D. If D replies with 0, return 0.
12. Return 1.

Game_7 : This game is defined identically to Game_6, except that matrix $\mathbf{M}''$ is again chosen as a rank 1 matrix. (meaning that $\widetilde{r}$ is again generated as the output of $\mathsf{SimCoin}$). Identically to the transition we made to move from Game_3 to Game_4, due to the μ-explicit hardness of DDH, we have

$$|\Pr[\mathsf{E}_6] - \Pr[\mathsf{E}_7]| \leq \mu(\lambda)$$

.
The description of the challenger is as follows:

1. $g \leftarrow_r \mathbb{G}$
2. $\mathbf{v}' \leftarrow_r \mathbb{Z}_p^{m+1} \backslash \{\mathbf{0}\}$
3. $\alpha_i \leftarrow_r \mathbb{Z}_p^*$, for all $i \in [m]$.

4. $\mathbf{w}'_i \leftarrow \alpha_i \mathbf{v}'$, for all $i \in [m]$.
5. $\mathbf{M}'' \leftarrow (\mathbf{v}'|\mathbf{w}'_1|\dots|\mathbf{w}'_m) \in \mathbb{Z}_p^{(m+1)\times(m+1)}$.
6. $\tilde{z} \leftarrow_r [1/\nu]$
7. $\tilde{z}' \leftarrow_r [1/\nu]$
8. $\mathbf{M} \leftarrow \mathbf{M}'' + \tilde{z}' \cdot \mathbf{I}_{m+1}$.
9. $r_{\mathsf{ZAP},i} \leftarrow_r \{0,1\}^{\ell'}$, for all $i \in [m]$
10. Set the simulated public-coin as $\widetilde{r} := ((r_{\mathsf{ZAP},i})_{i\in[m]}, g, g^{\mathbf{M}}) \in \{0,1\}^{\ell}$.
11. Set the extraction trapdoor as $\tau := ((\alpha_i)_{i\in[m]}, \tilde{z}')$.
12. Send $(m, \widetilde{r})$ to $\mathcal{A}$, which replies with $(\sigma, S, \rho_S^*, \{\pi_i\}_{i\in S}, \mathsf{st})$.
13. Parse $(H, g^s, z) \leftarrow \sigma$.
14. If $\tilde{z} \neq z$, return 0.
15. Send st to D. If D replies with 0, return 0.
16. Return 1.

By using the triangle inequality, we have therefore shown the following bound:

$$|\Pr[\mathsf{E}_0] - \Pr[\mathsf{E}_7]| \leq 2 \cdot \mu(\lambda) + m \cdot \varepsilon_{\mathsf{sound}} = \nu(\lambda) \cdot \mathsf{negl}(\lambda),$$

where the equality comes from our choice of $\varepsilon_{\mathsf{sound}}$ and $\nu(\lambda)$. Moreover, observe that in Game_7, $\tilde{z}$ is information theoretically hidden from $\mathcal{A}$ and D. Therefore, the probability of $\tilde{z} = z$ is ν regardless of all the other randomness. Namely, we have the following:

$$\begin{aligned}\Pr[\mathsf{E}_7] &= \Pr[\tilde{z} = z] \cdot \Pr\left[\begin{matrix}(\widetilde{r}, \tau) \leftarrow_r \mathsf{SimCoin}(1^\lambda, m) \\ (\sigma, S, \rho_S^*, \{\pi_i\}_{i\in S}, \mathsf{st}) \leftarrow_r \mathcal{A}(m, \widetilde{r})\end{matrix} : \ D(\mathsf{st}) = 1\right], \\ &= \nu(\lambda) \cdot \Pr\left[\begin{matrix}(\widetilde{r}, \tau) \leftarrow_r \mathsf{SimCoin}(1^\lambda, m) \\ (\sigma, S, \rho_S^*, \{\pi_i\}_{i\in S}, \mathsf{st}) \leftarrow_r \mathcal{A}(m, \widetilde{r})\end{matrix} : \ D(\mathsf{st}) = 1\right].\end{aligned}$$

Now notice that this corresponds to the probability in the right-hand side of the ν-successful extraction condition in Definition 14. Since $\Pr[\mathsf{E}_0]$ corresponded to the probability in the left-hand side, this completes the proof of the theorem. It only remains to prove Lemma 31 below.

Lemma 31. *If the* IHBG*-friendly ZAP for* $\mathscr{L}_{\mathsf{LPWW}}$ *is adaptively computational* $\varepsilon_{\mathsf{sound}}$*-sound w.r.t. colinear parameters, then we have* $|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_2]| \leq m \cdot \varepsilon_{\mathsf{sound}}$.

Proof. Observe that Game_1 and Game_2 will only differ in the event that $\mathsf{VerifyBit}(\widetilde{r}, \sigma, i^*, 1-\rho_{i^*}, \pi_{i^*}) = \top$ for some $i^* \in S$. Let us call this event F. Then we have $|\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_2]| \leq \Pr[\mathsf{F}]$. Below, we upper bound $\Pr[\mathsf{F}]$ by constructing an adversary $\mathcal{B}$, which runs $\mathcal{A}$ internally, against the adaptive computational $\varepsilon_{\mathsf{sound}}$-soundness w.r.t. colinear parameters of the IHBG-friendly ZAP for $\mathscr{L}_{\mathsf{LPWW}}$. The description of $\mathcal{B}$ follows:[14]

$\mathcal{B}$ receives a random language parameter $\mathsf{par} = (g^{\mathbf{v}'}, g^{\mathbf{w}'}) \in \mathsf{Col}(\mathbb{G})$ and a public-coin $r_{\mathsf{ZAP}} \in \{0,1\}^{\ell'}$ as the problem instance, where note that $\mathbf{w}' = \alpha \cdot \mathbf{v}'$ for some $\alpha \in \mathbb{Z}_p^*$.[15] First, $\mathcal{B}$ samples a random index $i^* \leftarrow_r [m]$ and sets $\mathsf{par}_{i^*} := \mathsf{par}$ and $r_{\mathsf{ZAP},i^*} := r_{\mathsf{ZAP}}$, which implicitly sets $\mathbf{w}_{i^*} := \mathbf{w}'$. It then further samples $r_{\mathsf{ZAP},i} \leftarrow_r \{0,1\}^{\ell'}$ and $\alpha_i \leftarrow_r \mathbb{Z}_p^*$ for all $i \in [m]\backslash\{i^*\}$ and sets $g^{\mathbf{w}'_i} = (g^{\mathbf{w}'})^{\alpha_i}$. Note that $\mathcal{B}$ can compute everything without knowledge of the secret exponents $(\mathbf{v}', \mathbf{w}')$. $\mathcal{B}$ then samples $\tilde{z} \leftarrow_r [1/\nu]$ and computes $g^{\mathbf{M}} = g^{\mathbf{M}''+\tilde{z}\cdot\mathbf{I}_{m+1}}$, where $\mathbf{M}''$ is implicitly set as $(\mathbf{v}'|\mathbf{w}'_1|\dots|\mathbf{w}'_m) \in \mathbb{Z}_p^{(m+1)\times(m+1)}$. $\mathcal{B}$ sets the simulated public-coin of the Π_{IHBG} as $\widetilde{r} = ((r_{\mathsf{ZAP},i})_{i\in[m]}, g^{\mathbf{M}})$, and invokes $\mathcal{A}$ on input $(m, \widetilde{r})$. When $\mathcal{A}$ outputs $(\sigma, S, \rho_S^*, \{\pi_i\}_{i\in S}, \mathsf{st})$, $\mathcal{B}$ parses $(H, g^s, z) \leftarrow \sigma$ and $(g^{u_{i^*}}, \pi_{\mathsf{ZAP},i^*}) \leftarrow \pi_{i^*}$, and outputs its forged statement-proof pair as $(x^* := (g^s, g^{u_{i^*}}), \pi_{\mathsf{ZAP}}^* := \pi_{\mathsf{ZAP},i^*})$.

We analyze the behavior of $\mathcal{B}$. First, since the given language parameter par is a random element over $\mathsf{Col}(\mathbb{G})$, $\mathcal{B}$ perfectly simulates the view of $\mathcal{A}$ in both Game_1 and Game_2. Next, we show $x^* = (g^s, g^{u_{i^*}}) \notin \mathscr{L}_{\mathsf{LPWW}}^{\mathsf{par}}$. By contradiction, assume $(g^s, g^{u_{i^*}}) \in \mathscr{L}_{\mathsf{LPWW}}^{\mathsf{par}}$. Namely, there exists some $\mathbf{y} \in \mathbb{Z}_p^d$ such that $s = \mathbf{y}^\top \mathbf{v}$ and $u_{i^*} = \mathbf{y}^\top \mathbf{w}$. Now, due to the definition of the Open algorithm, we have $\rho_{i^*} = H(g^{\alpha \cdot s})$. On the other hand, since $\mathsf{VerifyBit}(\widetilde{r}, \sigma, i^*, 1-\rho_{i^*}, \pi_{i^*}) = \top$, we have $1-\rho_{i^*} = H(g^{u_{i^*}})$. However, since $u_{i^*} = \mathbf{y}^\top \mathbf{w} = \alpha \cdot \mathbf{y}^\top \mathbf{v} = \alpha \cdot s$, the output of the hash function H cannot be different. Therefore, we have $x^* = (g^s, g^{u_{i^*}}) \notin \mathscr{L}_{\mathsf{LPWW}}^{\mathsf{par}}$. Then, since $i^* \in [m]$ is statistically hidden from $\mathcal{A}$,

[14] Note that $\mathcal{B}$ does not need to invoke D below since event F can be checked without the output of D.

[15] To be precise, we assume a random group generator $g \in \mathbb{G}$ is also provided to $\mathcal{B}$ as part of the challenge, which $\mathcal{B}$ uses to simulate the view of $\mathcal{A}$.

conditioning on event F occuring, we have $x^* \notin \mathscr{L}^{\mathsf{par}}_{\mathsf{LPWW}}$ and $\mathsf{ZAP.Verify}(\mathsf{par}, r_{\mathsf{ZAP}}, x^*, \pi^*_{\mathsf{ZAP}}) = \top$ with probability $1/m$. Therefore, due to the adaptive computational $\epsilon_{\mathsf{sound}}$-soundness w.r.t. colinear parameters of the IHBG-friendly ZAP, we obtain $\Pr[\mathsf{F}] \leq m \cdot \epsilon_{\mathsf{sound}}$. This concludes the proof.

Theorem 32 (Statistical Hiding). *If the* IHBG*-friendly ZAP for* $\mathscr{L}_{\mathsf{LPWW}}$ *is doubly-adaptive statistically witness indistinguishable, the hash function family* $\mathcal{H}$ *is universal, and* $\nu(\lambda)$ *is negligible, then* Π_{IHBG} *is statistically hiding.*

Proof. We first provide the description of the simulator Sim used to simulate the commitment σ and proof for opening $\{\pi_i\}_{i\in S}$ below:

$\mathsf{Sim}(m, r, S, \rho_S)$: On input a polynomial m, public-coin r, set S, and hidden-bits string ρ_S, it proceeds as follows:

1. Parse $((r_{\mathsf{ZAP},i})_{i\in[m]}, g, g^{\mathbf{M}}) \leftarrow r$.
2. $H \leftarrow_r \mathcal{H}$.
3. $z \leftarrow_r [1/\nu]$.
4. $g^{\mathbf{M}'} \leftarrow g^{\mathbf{M} - z\cdot \mathbf{I}_{m+1}}$.
5. Recover matrix $\mathbf{M}' := (\mathbf{v}'|\mathbf{w}'_1|\cdots|\mathbf{w}'_m) \in \mathbb{Z}_p^{(m+1)\times(m+1)}$ by brute force.
6. If $\mathbf{M}'$ is not full-rank, return $\perp$.
7. $\rho_i \leftarrow \{0,1\}^m$, for all $i \in [m]$.
8. Compute $Y := \{\mathbf{y}' \in \mathbb{Z}_p^{m+1} : H(g^{(\mathbf{y}')^\top \mathbf{w}'_i}) = \rho_i, \text{ for all } i \in S\}$ (inefficiently).
9. If $Y = \emptyset$, return 0.
10. $\mathbf{y} \leftarrow_r Y$.
11. $g^s \leftarrow g^{\mathbf{y}^\top \mathbf{v}'}$ and $g^{u_i} \leftarrow g^{\mathbf{y}^\top \mathbf{w}'_i}$, for all $i \in S$.
12. Set the language parameter $\mathsf{par}_i := (g^{\mathbf{v}'}, g^{\mathbf{w}'_i})$ and the statement $x_i := (g^s, g^{u_i})$ for the language $\mathscr{L}^{\mathsf{par}_i}_{\mathsf{LPWW}}$, for all $i \in S$.
13. $\pi_{\mathsf{ZAP},i} \leftarrow_r \mathsf{ZAP.Prove}(\mathsf{par}_i, r_{\mathsf{ZAP},i}, x_i, \mathbf{y})$, for all $i \in S$.
14. $\pi_i \leftarrow (g^{u_i}, \pi_{\mathsf{ZAP},i})$, for all $i \in S$.
15. $\sigma \leftarrow (H, g^s, z)$.
16. Return $(\sigma, \{\pi_i\}_{i\in S})$.

We consider the following sequence of games between an unbounded adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ and a challenger, where the first and last games corresponding respectively to the honest game using $\mathsf{GenBits}$ and simulated game using Sim of the statistical-hiding property in Definition 14. We denote E_i as the event that the challenger outputs 1 in Game_i.

Game_0**:** This game is the same as the real game where the challenger uses $\mathsf{GenBits}$ to generate the hidden-bits. By definition, we have

$$\Pr[\mathsf{E}_0] = \epsilon.$$

Below, we give the full description of the challenger, which receives as input the security parameter λ (in unary), polynomial m and public-coin r, and proceeds as follows:

1. Parse $((r_{\mathsf{ZAP},i})_{i\in[m]}, g, g^{\mathbf{M}}) \leftarrow r$.
2. $H \leftarrow_r \mathcal{H}$.
3. $z \leftarrow_r [1/\nu]$.
4. $g^{\mathbf{M}'} \leftarrow g^{\mathbf{M} - z\cdot \mathbf{I}_{m+1}}$.
5. $\mathbf{y} \leftarrow \mathbb{Z}_p^{m+1}$.
6. $g^s \leftarrow g^{\mathbf{y}^\top \mathbf{v}'}$ and $g^{u_i} \leftarrow g^{\mathbf{y}^\top \mathbf{w}'_i}$, for all $i \in S$, where $\mathbf{M} := (\mathbf{v}'|\mathbf{w}'_1|\cdots|\mathbf{w}'_m) \in \mathbb{Z}_p^{(m+1)\times(m+1)}$.
7. Set the language parameter $\mathsf{par}_i := (g^{\mathbf{v}'}, g^{\mathbf{w}'_i})$ and the statement $x_i := (g^s, g^{u_i})$ for the language $\mathscr{L}^{\mathsf{par}_i}_{\mathsf{LPWW}}$, for all $i \in S$.
8. $\pi_{\mathsf{ZAP},i} \leftarrow_r \mathsf{ZAP.Prove}(\mathsf{par}_i, r_{\mathsf{ZAP},i}, x_i, \mathbf{y})$, for all $i \in [m]$.
9. $\pi_i \leftarrow (g^{u_i}, \pi_{\mathsf{ZAP},i})$, for all $i \in [m]$.
10. $\sigma \leftarrow (H, g^s, z)$.
11. $\rho_i \leftarrow H(g^{u_i})$, for all $i \in [m]$.
12. Send $\rho := (\rho_i)_{i\in[m]}$ to $\mathcal{A}_0$, which replies with a set S. If $S \not\subseteq [m]$, return 0.
13. Send $(r, S, \rho, \sigma, \{\pi_i\}_{i\in S})$ to $\mathcal{A}_1$. If $\mathcal{A}_1$ replies with 1, return 1, otherwise, return 0.

Game_1 : This game is defined identically to Game_0, except that $\mathsf{GenBits}$ is modified as follows: Upon computing $g^{\mathbf{M}'} = g^{(\mathbf{v}'|\mathbf{w}'_1|\ldots|\mathbf{w}'_m)}$, the challenger recovers the discrete logarithms, obtaining $\mathbf{v}', \mathbf{w}'_1, \ldots, \mathbf{w}'_m$. Then, it picks $(s\|\mathbf{u}) \leftarrow_r \mathbb{Z}_p^{m+1}$ and computes $\mathbf{y} = \mathbf{M}'^{-1}(s\|\mathbf{u})$, where we show $\mathbf{M}'$ is invertible with overwhelming probability for our parameter choice ν and m. We later prove in Lemma 33 that we have

$$|\Pr[\mathsf{E}_0] - \Pr[\mathsf{E}_1]| \leq (m+1) \cdot \nu(\lambda).$$

The description of the challenger is as follows:

1. Parse $((r_{\mathsf{ZAP},i})_{i\in[m]}, g, g^{\mathbf{M}}) \leftarrow r$.
2. $H \leftarrow_r \mathcal{H}$.
3. $z \leftarrow_r [1/\nu]$.
4. $g^{\mathbf{M}'} \leftarrow g^{\mathbf{M}-z\cdot\mathbf{I}_{m+1}}$.
5. Recover $\mathbf{M}' := (\mathbf{v}'|\mathbf{w}'_1|\ldots|\mathbf{w}'_m) \in \mathbb{Z}_p^{(m+1)\times(m+1)}$ by brute force.
6. If $\mathbf{M}'$ is not full-rank, return 0.
7. $(s\|\mathbf{u}) \leftarrow_r \mathbb{Z}_p^{m+1}$.
8. Set the language parameter $\mathsf{par}_i := (g^{\mathbf{v}'}, g^{\mathbf{w}'_i})$ and the statement $x_i := (g^s, g^{u_i})$ for the language $\mathscr{L}^{\mathsf{par}_i}_{\mathsf{LPWW}}$, where u_i denotes the i-th entry of $\mathbf{u}$, for all $i \in [m]$.
9. $\mathbf{y} \leftarrow \mathbf{M}'^{-1}(s\|\mathbf{u})$.
10. $\pi_{\mathsf{ZAP},i} \leftarrow_r \mathsf{ZAP.Prove}(\mathsf{par}_i, r_{\mathsf{ZAP},i}, x_i, \mathbf{y})$, for all $i \in [m]$.
11. $\pi_i \leftarrow (g^{u_i}, \pi_{\mathsf{ZAP},i})$, for all $i \in [m]$.
12. $\sigma \leftarrow (H, g^s, z)$.
13. $\rho_i \leftarrow H(g^{u_i})$, for all $i \in [m]$.
14. Send $\rho := (\rho_i)_{i\in[m]}$ to $\mathcal{A}_0$, which replies with a set S. If $S \not\subseteq [m]$, return 0.
15. Send $(r, S, \rho, \sigma, \{\pi_i\}_{i\in S})$ to $\mathcal{A}_1$. If $\mathcal{A}_1$ replies with 1, return 1, otherwise, return 0.

Game_2 : This game is defined identically to Game_1, except that $\mathbf{u}$ is sampled differently. First, the challenger picks $\mathbf{u}'' \leftarrow_r \mathbb{Z}_p^m$, and then it samples $\mathbf{u} \leftarrow_r \{\mathbf{u}' \in \mathbb{Z}_p^m : H(g^{u'_i}) = H(g^{u''_i}), \text{ for all } i \in S\}$, which it can do since computing the set of all valid $\mathbf{u}'$ can be done inefficiently. For completeness, we later formally prove in Lemma 34 that we have

$$\Pr[\mathsf{E}_1] = \Pr[\mathsf{E}_2].$$

The description of the challenger is as follows:

1. Parse $((r_{\mathsf{ZAP},i})_{i\in[m]}, g, g^{\mathbf{M}}) \leftarrow r$.
2. $H \leftarrow_r \mathcal{H}$.
3. $z \leftarrow_r [1/\nu]$.
4. $g^{\mathbf{M}'} \leftarrow g^{\mathbf{M}-z\cdot\mathbf{I}_{m+1}}$.
5. Recover $\mathbf{M}' := (\mathbf{v}'|\mathbf{w}'_1|\ldots|\mathbf{w}'_m) \in \mathbb{Z}_p^{(m+1)\times(m+1)}$ by brute force.
6. If $\mathbf{M}'$ is not full-rank, return 0.
7. $(s\|\mathbf{u}'') \leftarrow_r \mathbb{Z}_p^{m+1}$.
8. $\rho_i \leftarrow H(g^{u''_i})$, for all $i \in [m]$, where u''_i denotes the i-th entry of $\mathbf{u}''$.
9. Send $\rho := (\rho_i)_{i\in[m]}$ to $\mathcal{A}_0$, which replies with a set S. If $S \not\subseteq [m]$, return 0.
10. Compute $U = \{\mathbf{u}' \in \mathbb{Z}_p^m : H(g^{u'_i}) = \rho_i, \text{ for all } i \in S\}$ (inefficiently).
11. $\mathbf{u} \leftarrow_r U$.
12. Set the language parameter $\mathsf{par}_i := (g^{\mathbf{v}'}, g^{\mathbf{w}'_i})$ and the statement $x_i := (g^s, g^{u_i})$ for the language $\mathscr{L}^{\mathsf{par}_i}_{\mathsf{LPWW}}$ for all $i \in [m]$.
13. $\mathbf{y} \leftarrow \mathbf{M}'^{-1}(s\|\mathbf{u})$.
14. $\pi_{\mathsf{ZAP},i} \leftarrow_r \mathsf{ZAP.Prove}(\mathsf{par}_i, r_{\mathsf{ZAP},i}, x_i, \mathbf{y})$, for all $i \in [m]$.
15. $\pi_i \leftarrow (g^{u_i}, \pi_{\mathsf{ZAP},i})$, for all $i \in [m]$.
16. $\sigma \leftarrow (H, g^s, z)$.
17. Send $(r, S, \rho, \sigma, \{\pi_i\}_{i\in S})$ to $\mathcal{A}_1$. If $\mathcal{A}_1$ replies with 1, return 1, otherwise, return 0.

Game_3 : This game is defined identically to Game_2, except for the hidden-bits generation procedure, which now first samples $\rho \leftarrow_r \{0,1\}^m$ and then uniformly randomly picks $\mathbf{u}$ such that $H(g^{u_i}) = \rho_i$ for all $i \in S$, if such $\mathbf{u}$ exists. We later prove in Lemma 35 that, due to the statistical uniformity of the hash function, we have

$$|\Pr[\mathsf{E}_2] - \Pr[\mathsf{E}_3]| \leq \mathsf{negl}(\lambda).$$

The description of the challenger is as follows:

1. Parse $((r_{\mathsf{ZAP},i})_{i\in[m]}, g, g^{\mathbf{M}}) \leftarrow r$.
2. $H \leftarrow_r \mathcal{H}$.
3. $z \leftarrow_r [1/\nu]$.
4. $g^{\mathbf{M}'} \leftarrow g^{\mathbf{M}-z\cdot\mathbf{I}_{m+1}}$.
5. Recover $\mathbf{M}' := (\mathbf{v}'|\mathbf{w}'_1|\ldots|\mathbf{w}'_m) \in \mathbb{Z}_p^{(m+1)\times(m+1)}$ by brute force.
6. If $\mathbf{M}'$ is not full-rank, return 0.
7. $\rho_i \leftarrow \{0,1\}^m$, for all $i \in [m]$.
8. Send $\rho := (\rho_i)_{i\in[m]}$ to $\mathcal{A}_0$, which replies with a set S. If $S \not\subseteq [m]$, return 0.
9. Compute $U = \{\mathbf{u}' \in \mathbb{Z}_p^m : H(g^{u'_i}) = \rho_i, \text{ for all } i \in S\}$ (inefficiently).
10. If $U = \emptyset$, return 0.
11. $\mathbf{u} \leftarrow_r U$.
12. $s \leftarrow_r \mathbb{Z}_p$.
13. Set the language parameter $\mathsf{par}_i := (g^{\mathbf{v}'}, g^{\mathbf{w}'_i})$ and the statement $x_i := (g^s, g^{u_i})$ for the language $\mathscr{L}^{\mathsf{par}_i}_{\mathsf{LPWW}}$, for all $i \in [m]$.
14. $\mathbf{y} \leftarrow \mathbf{M}'^{-1}(s\|\mathbf{u})$.
15. $\pi_{\mathsf{ZAP},i} \leftarrow_r \mathsf{ZAP.Prove}(\mathsf{par}_i, r_{\mathsf{ZAP},i}, x_i, \mathbf{y})$, for all $i \in S$.
16. $\pi_i \leftarrow (g^{u_i}, \pi_{\mathsf{ZAP},i})$, for all $i \in [m]$.
17. $\sigma \leftarrow (H, g^s, z)$.
18. Send $(r, S, \rho, \sigma, \{\pi_i\}_{i\in S})$ to $\mathcal{A}_1$. If $\mathcal{A}_1$ replies with 1, return 1, otherwise, return 0.

Game_4 : This game is defined identically to Game_3, except that it only samples the necessary entries in $\mathbf{u}$ and samples an additional vector $\mathbf{y}^r$ uniformly at random, conditioned on the fact that $g^{(\mathbf{y}^r)^\top\mathbf{v}'} = g^s$ and $g^{(\mathbf{y}^r)^\top\mathbf{w}'_i} = g^{u_i}$, for all $i \in S$. Then, the proofs $\{\pi_{\mathsf{ZAP},i}\}_{i\in S}$ are computed using witness $\mathbf{y}^r$ instead of witness $\mathbf{y}$. We later prove in Lemma 36 that, due to the doubly-adaptive statistical witness indistinguishability of the IHBG-friendly ZAP for $\mathscr{L}_{\mathsf{LPWW}}$, we have

$$|\Pr[\mathsf{E}_3] - \Pr[\mathsf{E}_4]| \leq \mathsf{negl}(\lambda).$$

The description of the challenger is as follows, where note that $Y_{s,\mathbf{u}} \neq \emptyset$ is guaranteed in case the challenger hasn't returned 0:

1. Parse $((r_{\mathsf{ZAP},i})_{i\in[m]}, g, g^{\mathbf{M}}) \leftarrow r$.
2. $H \leftarrow_r \mathcal{H}$.
3. $z \leftarrow_r [1/\nu]$.
4. $g^{\mathbf{M}'} \leftarrow g^{\mathbf{M}-z\cdot\mathbf{I}_{m+1}}$.
5. Recover $\mathbf{M}' := (\mathbf{v}'|\mathbf{w}'_1|\ldots|\mathbf{w}'_m) \in \mathbb{Z}_p^{(m+1)\times(m+1)}$ by brute force.
6. If $\mathbf{M}'$ is not full-rank, return 0.
7. $\rho_i \leftarrow \{0,1\}^m$, for all $i \in [m]$.
8. Send $\rho := (\rho_i)_{i\in[m]}$ to $\mathcal{A}_0$, which replies with a set S. If $S \not\subseteq [m]$, return 0.
9. Compute $U = \{\mathbf{u}' \in \mathbb{Z}_p^S : H(g^{u'_i}) = \rho_i, \text{ for all } i \in S\}$ (inefficiently), where $\mathbf{x} \in \mathbb{Z}_p^S$ is a vector of length $|S|$ indexed by the elements in the set S
10. If $U = \emptyset$, return 0.
11. $\mathbf{u} \leftarrow_r U$.
12. $s \leftarrow_r \mathbb{Z}_p$.
13. Set the language parameter $\mathsf{par}_i := (g^{\mathbf{v}'}, g^{\mathbf{w}'_i})$ and the statement $x_i := (g^s, g^{u_i})$ for the language $\mathscr{L}^{\mathsf{par}_i}_{\mathsf{LPWW}}$, for all $i \in S$.
14. Compute $Y_{s,\mathbf{u}} := \{\mathbf{y}' \in \mathbb{Z}_p^{m+1} : g^{(\mathbf{y}')^\top\mathbf{v}'} = g^s \wedge g^{(\mathbf{y}')^\top\mathbf{w}'_i} = g^{u_i}, \text{ for all } i \in S\}$ (inefficiently).
15. $\mathbf{y}^r \leftarrow_r Y_{s,\mathbf{u}}$.
16. $\pi_{\mathsf{ZAP},i} \leftarrow_r \mathsf{ZAP.Prove}(\mathsf{par}_i, r_{\mathsf{ZAP},i}, x_i, \mathbf{y}^r)$, for all $i \in S$.
17. $\pi_i \leftarrow (g^{u_i}, \pi_{\mathsf{ZAP},i})$, for all $i \in S$.
18. $\sigma \leftarrow (H, g^s, z)$.
19. Send $(r, S, \rho, \sigma, \{\pi_i\}_{i\in S})$ to $\mathcal{A}_1$. If $\mathcal{A}_1$ replies with 1, return 1, otherwise, return 0.

Game_5 This game is defined identically to Game_4, except that it now first samples $\mathbf{y}$ uniformly at random, conditioned on the fact that $H(g^{\mathbf{y}^\top\mathbf{w}'_i}) = \rho_i$, for all $i \in S$. It then computes $g^s = g^{\mathbf{y}^\top\mathbf{v}'}$ and $g^{u_i} = g^{\mathbf{y}^\top\mathbf{w}'_i}$ for all $i \in S$. Namely, the order of operations is switched: in Game_4, first $\mathbf{u}$ and s are sampled and $\mathbf{y}$ is determined afterwards, while in Game_5, the challenger first samples $\mathbf{y}$ and only after it can compute $\mathbf{u}$ and s accordingly. We later prove in Lemma 37 that we have

$$\Pr[\mathsf{E}_4] = \Pr[\mathsf{E}_5].$$

The description of the challenger is as follows:

1. Parse $((r_{\mathsf{ZAP},i})_{i\in[m]}, g, g^{\mathbf{M}}) \leftarrow r$.
2. $H \leftarrow_r \mathcal{H}$.
3. $z \leftarrow_r [1/\nu]$.
4. $g^{\mathbf{M}'} \leftarrow g^{\mathbf{M}-z\cdot\mathbf{I}_{m+1}}$.
5. Recover $\mathbf{M}' := (\mathbf{v}'|\mathbf{w}'_1|\dots|\mathbf{w}'_m) \in \mathbb{Z}_p^{(m+1)\times(m+1)}$ by brute force.
6. If $\mathbf{M}'$ is not full-rank, return 0.
7. $\rho_i \leftarrow \{0,1\}^m$, for all $i \in [m]$.
8. Send $\rho := (\rho_i)_{i\in[m]}$ to $\mathcal{A}_0$, which replies with a set S. If $S \not\subseteq [m]$, return 0.
9. Compute $Y := \{\mathbf{y}' \in \mathbb{Z}_p^{m+1} : H(g^{(\mathbf{y}')^\top \mathbf{w}'_i}) = \rho_i, \text{ for all } i \in S\}$ (inefficiently).
10. If $Y = \emptyset$, return 0.
11. $\mathbf{y} \leftarrow_r Y$.
12. $g^s \leftarrow g^{\mathbf{y}^\top \mathbf{v}'}$ and $g^{u_i} \leftarrow g^{\mathbf{y}^\top \mathbf{w}'_i}$, for all $i \in S$.
13. Set the language parameter $\mathsf{par}_i := (g^{\mathbf{v}'}, g^{\mathbf{w}'_i})$ and the statement $x_i := (g^s, g^{u_i})$ for the language $\mathscr{L}^{\mathsf{par}_i}_{\mathsf{LPWW}}$, for all $i \in S$.
14. $\pi_{\mathsf{ZAP},i} \leftarrow_r \mathsf{ZAP.Prove}(\mathsf{par}_i, r_{\mathsf{ZAP},i}, x_i, \mathbf{y})$, for all $i \in S$.
15. $\pi_i \leftarrow (g^{u_i}, \pi_{\mathsf{ZAP},i})$, for all $i \in S$.
16. $\sigma \leftarrow (H, g^s, z)$.
17. Send $(r, S, \rho, \sigma, \{\pi_i\}_{i\in S})$ to $\mathcal{A}_1$. If $\mathcal{A}_1$ replies with 1, return 1, otherwise, return 0.

Finally, notice that Game_5 is equivalent to first sampling $\rho \leftarrow_r \{0,1\}^m$, running $S \leftarrow_r \mathcal{A}_0(\rho)$, $(\sigma, \{\pi_i\}_{i\in S}) \leftarrow_r \mathsf{Sim}(m, r, S, \rho_S)$, and checking whether $S \subseteq [m]$ and $\mathcal{A}_1(r, S, \rho, \sigma, \{\pi_i\}_{i\in S}) = 1$. In particular, Game_5 corresponds to the simulated game using Sim of the statistical-hiding property. Combining all the arguments and using the triangle inequality, we have therefore shown the following bound:

$$|\Pr[\mathsf{E}_0] - \Pr[\mathsf{E}_5]| \leq (m+1)\cdot\nu(\lambda) + \mathsf{negl}(\lambda) = \mathsf{negl}(\lambda),$$

where the equality comes from the fact that m is polynomial and ν is negligible. This completes the proof of the theorem. It remains to prove Lemmas 33 to 37.

Lemma 33. *We have* $|\Pr[\mathsf{E}_0] - \Pr[\mathsf{E}_1]| \leq (m+1)\cdot\nu(\lambda)$.

Proof. The only difference between the two games occurs when the matrix $\mathbf{M}'$ is not full-rank. Namely, when matrix $\mathbf{M}'$ is full-rank, this means that $\mathbf{M}'$ and $\mathbf{M}'^{-1}$ correspond to bijective linear functions. Therefore, since $(s\|\mathbf{u}) \leftarrow_r \mathbb{Z}_p^{m+1}$ is uniformly random, vector $\mathbf{y}$ is also uniformly distributed in $\mathbb{Z}_p^{m+1}$ and the distributions in Game_0 and Game_1 are identical.

It remains to show that the matrix $\mathbf{M}'$ is full-rank with overwhelming probability $1-(m+1)\cdot\nu$. Assume that $\mathbf{M}' = \mathbf{M} - z\cdot\mathbf{I}_{m+1}$ was not full-rank, then there exists a vector $\mathbf{t}$ in the right kernel of $\mathbf{M}'$, meaning that $\mathbf{M}'\mathbf{t} = \mathbf{0}$. But then, $\mathbf{M}\mathbf{t} = z\cdot\mathbf{t}$, which means that $\mathbf{t}$ is an eigenvector and z is a eigenvalue of $\mathbf{M}$. Matrix $\mathbf{M}$ has at most $m+1$ eigenvalues. Since $z \leftarrow_r [1/\nu]$ is uniformly random, the probability that z is an eigenvalue is at most $(m+1)\cdot\nu$ (which happens in the worse-case scenario in which all $m+1$ eigenvalues of $\mathbf{M}$ are in the set $[1/\nu]$).

Lemma 34. *We have* $\Pr[\mathsf{E}_1] = \Pr[\mathsf{E}_2]$.

Proof. The distributions of $\mathbf{u}$ in Game_1 and in Game_2 are identical. The only difference is that in In Game_2, the sampling of $\mathbf{u}$ is performed in two stages: the challenger first picks a vector $\mathbf{u}'' \leftarrow_r \mathbb{Z}_p^m$, computes $\rho_i \leftarrow H(g^{u''_i})$, and after sending ρ to $\mathcal{A}_0$, it receives the set S. Afterwards, the challenger samples $\mathbf{u} \leftarrow_r \{\mathbf{u}' \in \mathbb{Z}_p^m : H(g^{u'_i}) = \rho_i, \text{ for all } i \in S\}$. It is clear that the resulting distribution of $\mathbf{u}$ is identical to those of Game_1.

Lemma 35. *We have that* $|\Pr[\mathsf{E}_2] - \Pr[\mathsf{E}_3]| \leq \mathsf{negl}(\lambda)$ *assuming the hash function family* $\mathcal{H}$ *is universal.*

Proof. Notice that Game_2 and Game_3 are identical conditioned on the set U in Game_3 being non-empty. Therefore, we upper bound the probability of U being empty in Game_3. For a fixed $H \in \mathcal{H}$ and $\rho_1 \in \{0,1\}$, define the set $U_1 := \{u' \in \mathbb{Z}_p^m : H(g^{u'}) = \rho_1\}$. Then, by the union bound, we have $\Pr_{H,\rho_1,\cdots,\rho_m}[U \text{ is empty}] \leq m\cdot\Pr_{H,\rho_1}[U_1 \text{ is empty}]$. Finally, due to Lemma 2 and $p > \lambda^{\omega(1)}$, we have $\Pr_{H,\rho_1}[U_1 \text{ is empty}] \leq \mathsf{negl}(\lambda)$, hence completing the proof.

Lemma 36. *If the* IHBG*-friendly ZAP for* $\mathscr{L}_{\mathsf{LPWW}}$ *is doubly-adaptive statistically witness indistinguishable, then we have* $|\Pr[\mathsf{E}_3] - \Pr[\mathsf{E}_4]| \leq \mathsf{negl}(\lambda)$.

Proof. We consider $(m+1)$ intermediate games $\mathsf{Game}_3^0, \cdots, \mathsf{Game}_3^m$, where the challenger in Game_3^i generates the proof $\pi_{\mathsf{ZAP},j}$ for $j \in [i+1:m] \cap S$ using witness $\mathbf{y}$ as in Game_3 and the proofs for $j \in [i] \cap S$ using witness $\mathbf{y}^r$ as in Game_4. Noticing that we do not need to generate any proofs for $j \in [m] \backslash S$, we have $\mathsf{Game}_3^0 = \mathsf{Game}_3$ and $\mathsf{Game}_3^m = \mathsf{Game}_4$. We prove that the output distribution of $\mathcal{A}$ changes negligibly by transitioning from $\mathsf{Game}_3^{i^*}$ to $\mathsf{Game}_3^{i^*+1}$ for all $i^* \in [0:m-1]$.

Let $\mathcal{B}$ be an adversary against the doubly-adaptive statistically witness indistinguishability of the IHBG-friendly ZAP that internally runs $\mathcal{A}$ defined as follows: On input the security parameter in unary 1^λ, $\mathcal{B}$ runs identically up to Item 13 as the challenger in Game_3. It then computes $\mathbf{y}$ and $\mathbf{y}^r$ as in Item 14 in Game_3 and Item 14 in Game_4, respectively. $\mathcal{B}$ generates $\pi_{\mathsf{ZAP},i}$ using witness $\mathbf{y}$ for $i \in [i^*+2:m] \cap S$ and using witness $\mathbf{y}^r$ for $i \in [i^*] \cap S$. $\mathcal{B}$ then sets st as all its internal state and sends $(r_{\mathsf{ZAP},i^*}, \mathsf{par}_{i^*}, x_{i^*}, \mathbf{y}, \mathbf{y}^r, \mathsf{st})$ to the challenger of the IHBG-friendly ZAP witness indistinguishability game, and receives back π^*_{ZAP}. Finally, $\mathcal{B}$ sets $\pi_{\mathsf{ZAP},i^*+1} := \pi^*_{\mathsf{ZAP}}$ if $i^* \in S$ and simulates the rest of the challenger in Game_3 and outputs whatever output by $\mathcal{A}_1$. It can be checked that $\mathcal{B}$ perfectly simulates $\mathsf{Game}_3^{i^*}$ and $\mathsf{Game}_3^{i^*+1}$ when witnesses $\mathbf{y}$ and $\mathbf{y}^r$ are used to create π^*_{ZAP}, respectively. Therefore, by a standard hybrid argument over m-games, we conclude the proof.

Lemma 37. *We have* $\Pr[\mathsf{E}_4] = \Pr[\mathsf{E}_5]$.

Proof. It suffices to prove that the distribution of $(\mathbf{u}, s, \mathbf{y})$ in Game_4 and Game_5 are identical. First, we focus on the case that $\mathbf{M}'$ is full-rank since otherwise the challenger implicitly sets $(\mathbf{u}, s, \mathbf{y}) = (\bot, \bot, \bot)$ in both games. Conditioning on $\mathbf{M}'$ being full-rank, we have the set U in Game_4 is empty if and only if the set Y in Game_5 is empty. Therefore, we can further focus on the case that U in Game_4 is non-empty. Now, notice that since $\mathbf{M}$ is full-rank, the sets $\{Y_{s,\mathbf{u}}\}_{(s,\mathbf{u}) \in \mathbb{Z}_p \times U}$ are disjoint and all have the same size for a fixed choice of $H, \mathbf{M}, S$, and ρ. Therefore, since Y in Game_5 is identical to $\bigcup_{(s,\mathbf{u}) \in \mathbb{Z}_p \times U} Y_{s,\mathbf{u}}$, the distribution of first sampling $(s, \mathbf{u}) \leftarrow_r \mathbb{Z}_p \times U$ and sampling $\mathbf{y} \leftarrow_r Y_{s,\mathbf{u}}$ is identical to first sampling $\mathbf{y} \leftarrow_r Y$ and then setting $(s, (u_i)_{i \in S}) = (\mathbf{y}^\top \mathbf{v}', (\mathbf{y}^\top \mathbf{w}'_i)_{i \in S})$. We complete the proof by taking the probability over the other variables.

6 IHBG-Friendly Statistical ZAPs for $\mathscr{L}_{\mathsf{LPWW}}$

In this section, we provide two instantiations for the IHBG-friendly statistical ZAP used in the construction of IHBG from the previous section, one in pairing groups, and one in pairing-free groups. These constructions and their analysis constitute the proofs of Lemma 23 and Lemma 24.

6.1 First Construction: a Statistical ZAP for $\mathscr{L}_{\mathsf{LPWW}}$ in Pairing Groups

For this construction, we employ the Couteau-Hartmann compiler from [8]. The high-level idea of the compiler is very simple: assume that the family of parametrized languages $\mathscr{L}_{\mathsf{LPWW}} = \{\mathscr{L}^{\mathsf{par}}_{\mathsf{LPWW}}\}_{\mathsf{par} \in \Lambda}$ is defined over a group $\mathbb{G}_1$, such that there exists another group $\mathbb{G}_2$ and an asymmetric pairing from $\mathbb{G}_1 \times \mathbb{G}_2$ to a target group $\mathbb{G}_\mathsf{T}$. Let $g_2 \in \mathbb{G}_2$ be a generator of $\mathbb{G}_2$. Then, the Couteau-Hartmann compiler converts a Σ-protocol with linear answer for the target language into a statistical ZAP by parsing the random message of the verifier as a pair (g_2, g_2^e), where e is seen as some random verifier challenge for the Σ-protocol. The compiled ZAP is constructed by computing the first flow of the Σ-protocol normally, and the last flow (which is a linear function of the challenge e with coefficients known to the prover) "in the exponent of g_2" using (g_2, g_2^e). The verification step is carried out using a pairing. Below, we adapt this compiler to the family of parameterized languages $\mathscr{L}_{\mathsf{LPWW}}$ and prove its security.

Construction. Let $(\mathbb{G}_1, \mathbb{G}_2)$ be elliptic curves equipped with an asymmetric pairing $\bullet : \mathbb{G}_1 \times \mathbb{G}_e \mapsto \mathbb{G}_\mathsf{T}$, where $\mathbb{G}_1$ and $\mathbb{G}_2$ both have prime order p. We extend the definition of $\bullet$ to vectors in the conventional manner. Let g_1 be a generator of $\mathbb{G}_1$ and d be a vector length parameter. Let $\mathsf{par} = (\mathbf{g}, \mathbf{h}) \in \Lambda = (\mathbb{G}_1^d \backslash \{\mathbf{1}\})^2$ be the language parameters. We will rely on the Σ-protocol from Section 4.4 with repetition parameter $n = 1$. In particular, we do not require to rely on the adaptive soundness of the Σ-protocol (i.e., Lemma 29) to achieve adaptive soundness (looking ahead, higher value of n (i.e., adaptive soundness of the Σ-protocol) will only be useful in our pairing-free instantiation). The

construction of a ZAP for $\mathscr{L}_{\mathsf{LPWW}}$ over $\mathbb{G}_1$ with public coin length $\ell = 2\lceil \log |\mathbb{G}_2| \rceil$, denoted as Π_{ZAP}, is described as follows.

- $\mathsf{ZAP.Prove}(\mathsf{par}, r', x, w)$: On input parameters $\mathsf{par} = (\mathbf{g}, \mathbf{h}) \in \Lambda$, a public coin $r \in \{0,1\}^\ell$, a statement $x := (X, Y) \in \mathscr{L}^{\mathsf{par}}_{\mathsf{LPWW}}$ and a witness $w := \mathbf{y} \in \mathbb{Z}_p^d$ such that $(X, Y) = ((\mathbf{g}^\top)^{\mathbf{y}}, (\mathbf{h}^\top)^{\mathbf{y}})$, parse r as $(g_2, g_2^e) \in \mathbb{G}_2^2$ and proceed as follows:
  - Pick $\mathbf{z} \leftarrow_r \mathbb{Z}_p^d$ and set $(R, S) \leftarrow ((\mathbf{g}^\top)^{\mathbf{z}}, (\mathbf{h}^\top)^{\mathbf{z}})$. Note that this corresponds to computing the first flow of the prover in the Σ-protocol from Section 4.4, with $n = 1$.
  - Set $g_2^{\mathbf{d}} \leftarrow (g_2^e)^{\mathbf{y}} \cdot g_2^{\mathbf{z}}$. Note that this corresponds to computing the last flow of the prover in the Σ-protocol from Section 4.4, *in the exponent domain* of $\mathbb{G}_2$.
  - Output $\pi_{\mathsf{ZAP}} = (R, S, g_2^{\mathbf{d}})$.
- $\mathsf{ZAP.Verify}(\mathsf{par}, r, x, \pi_{\mathsf{ZAP}})$: On input parameters $\mathsf{par} = (\mathbf{g}, \mathbf{h}) \in \Lambda$, a public coin $r \in \{0,1\}^\ell$, a statement $x = (X, Y)$, and a proof π_{ZAP}, parse π_{ZAP} as $(R, S, g_2^{\mathbf{d}})$, and parse r as $(g_2, g_2^e) \in \mathbb{G}_2^2$. Check that $\mathbf{g}^\top \bullet g_2^{\mathbf{d}} = (X \bullet g_2^e) \cdot (R \bullet g_2)$ and $\mathbf{h}^\top \bullet g_2^{\mathbf{d}} = (Y \bullet g_2^e) \cdot (S \bullet g_2)$. Note that this corresponds to executing the verification procedure of the Σ-protocol from Section 4.4 (with $n = 1$), but using the pairings to emulate the exponentiations of $(\mathbf{g}^\top, \mathbf{h}^\top)$ and (X, Y) (which are all over $\mathbb{G}_1$) by $\mathbf{d}$ and e respectively, since the latter are now only known in the exponent of g_2.

Completeness follows directly from the completeness of the underlying Σ-protocol.

Lemma 38. *For any parametrized language $\mathscr{L}^{\mathsf{par}}_{\mathsf{LPWW}}$ over $\mathbb{G}_1$, Π_{ZAP} satisfies doubly-adaptive perfect witness indistinguishability.*

Proof. This follows directly from the perfect witness indistinguishability of the underlying Σ-protocol. Indeed, if there exists a (possibly) inefficient cheating adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ against the doubly-adaptive perfect witness indistinguishability of the above ZAP, then it can be lifted to an attack against the perfect witness indistinguishability of the Σ-protocol as follows: first run $\mathcal{A}_0$ to get the parameters $\mathsf{par} \in \Lambda$, the first flow $r = (g_2, g_2^e) \in \mathbb{G}_2^2$, and statement-witnesses pair (x, w_0, w_1), and state st. Then, extract $e \in \mathbb{Z}_p^*$ by brute force search of the discrete logarithm and send par and (x, w_0) (and (x, w_1)) to the prover in the Σ-protocol. Upon receiving the first flow (R, S) of the prover, send the challenge e. Upon receiving the last flow $\mathbf{d}$, lift it to the exponent of g_2 and feed $\mathcal{A}_1$ with $(R, S, g_2^{\mathbf{d}}, \mathsf{st})$. Output whatever $\mathcal{A}_1$ output; it is immediate to check that this translate to an attack against the witness indistinguishability of the Σ-protocol with exactly the same advantage as $\mathcal{A}$. □

Lemma 39. *For a randomly sampled parametrized language $\mathscr{L}^{\mathsf{par}}_{\mathsf{LPWW}}$ over $\mathbb{G}_1$, Π_{ZAP} satisfies adaptive computational $\varepsilon_{\mathsf{sound}}$-soundness w.r.t. colinear parameters $\mathsf{par} \in \mathsf{Col}(\mathbb{G}_1^d)$ under the explicit $\varepsilon_{\mathsf{sound}}$-hardness of the kernel Diffie-Hellman assumption in $\mathbb{G}_2$.*

Proof. Let $\mathcal{A}$ be an efficient adversary against the adaptive computational $\varepsilon_{\mathsf{sound}}$-soundness w.r.t. colinear parameters of Π_{ZAP}:

$$\Pr[\mathsf{par} \leftarrow_r \mathsf{Col}(\mathbb{G}_1^d), r \leftarrow \{0,1\}^\ell, (x, \pi_{\mathsf{ZAP}}) \leftarrow_r \mathcal{A}(\mathsf{par}, r) : x \notin \mathscr{L}^{\mathsf{par}}_{\mathsf{LPWW}} \wedge \mathsf{Verify}(\mathsf{par}, r, x, \pi_{\mathsf{ZAP}}) = \top] > \varepsilon_{\mathsf{sound}}.$$

We build an adversary against the explicit $\varepsilon_{\mathsf{sound}}$-hardness of the kernel Diffie-Hellman assumption in $\mathbb{G}_2$ as follows: the reduction receives $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathsf{T}, p, g_1, g_2, g_2^e)$ from the challenger for the kernel Diffie-Hellman assumption. It samples par by picking $(\mathbf{v}, s) \leftarrow_r (\mathbb{Z}_p^*)^d \times \mathbb{Z}_p^*$ and setting $\mathsf{par} \leftarrow (\mathbf{g}, \mathbf{h}) = (g_1^{\mathbf{v}}, g_1^{s \cdot \mathbf{v}})$, and sets r to be (g_2, g_2^e). Note that par is perfectly distributed as a random sample from $\mathsf{Col}(\mathbb{G}_1^d)$, and r is perfectly distributed as in Π_{ZAP}. Then, it gets $(x, \pi_{\mathsf{ZAP}}) \leftarrow \mathcal{A}(\mathsf{par}, r)$ from the ZAP adversary $\mathcal{A}$, and parses x as (X, Y) and π_{ZAP} as $(R, S, g_2^{\mathbf{d}})$. By definition of $\mathcal{A}$ and since par and r are distributed exactly as in the soundness game, it holds with probability at least $\varepsilon_{\mathsf{sound}}$ that $(X, Y) \notin \mathscr{L}^{\mathsf{par}}_{\mathsf{LPWW}}$ and $\mathsf{Verify}(\mathsf{par}, r, x, \pi_{\mathsf{ZAP}}) = \top$, i.e. we have the following three equations:

- $X^s / Y \neq g_1^0$ (this is equivalent to $(X, Y) \notin \mathscr{L}^{\mathsf{par}}_{\mathsf{LPWW}}$),
- $\mathbf{g}^\top \bullet g_2^{\mathbf{d}} = (X \bullet g_2^e) \cdot (R \bullet g_2)$, and
- $\mathbf{h}^\top \bullet g_2^{\mathbf{d}} = (Y \bullet g_2^e) \cdot (S \bullet g_2)$.

Now, since $\mathbf{h} = \mathbf{g}^s$, and using the bilinearity of the pairing, we get from the last two equations that

$$1 = (X^s / Y \bullet g_2^e) \cdot (R^s / S \bullet g_2) = (R^s / S, X^s / Y) \bullet (g_2, g_2^e)^\top$$

and therefore the vector $(R^s / S, X^s / Y)$, which the reduction can efficiently compute, is of the form (g_1^u, g_2^v) where (u, v) is in the kernel of $(1, e)$ and $v \neq 0$ (since $X^s / Y \neq g_1^0$); hence, the reduction outputs $(R^s / S, X^s / Y)$ and breaks the kernel Diffie-Hellman assumption with probability at least $\varepsilon_{\mathsf{sound}}$. □

This concludes the proof of Lemma 23. Plugging this IHBG-friendly adaptive statistical ZAP for $\mathscr{L}_{\mathsf{LPWW}}$ into the construction of IHBG of Section 5 and combining it with the construction of statistical ZAP for NP from any IHBG from Section 3, we get our first main theorem:

Theorem 40 (Statistical ZAPs in Pairing Groups). *Assume that the explicit μ-hardness of the DDH assumption holds in a group $\mathbb{G}_1$, and the explicit (μ/m)-hardness of the kernel Diffie-Hellman assumption holds in a group $\mathbb{G}_2$, where $(\mathbb{G}_1, \mathbb{G}_2)$ are groups equipped with a bilinear pairing, m is the output length of the* IHBG *protocol, and for any negligible function μ (which can be arbitrarily close to an inverse polynomial function). Then there exists an adaptive statistically witness indistinguishable ZAP for* NP *with non-adaptive computational soundness.*

As we mentioned in the preliminaries, the above theorem can be easily generalized to hold under any of the more general family of Diffie-Hellman assumption, such as the (decisional and kernel) matrix Diffie-Hellman assumption [15, 38] (which includes for example the k-Lin assumption from [27]).

6.2 Second Construction: a Statistical ZAP for $\mathscr{L}_{\mathsf{LPWW}}$ in Pairing-Free Groups

A Correlation-Intractable Hash Function for $\mathcal{R}_{\mathsf{LPWW}}$. Let λ be the security parameter. We consider a group $\tilde{\mathbb{G}}$ of order $q(\lambda)$ with $\lceil \log q \rceil \approx \lambda$. Let $\mathsf{Trunc} : \tilde{\mathbb{G}} \mapsto \{0,1\}^{\lambda/2}$ be the function which, on input a group element $\tilde{G} \in \tilde{\mathbb{G}}$, parses it as a $\lceil \log q \rceil$-bit string and returns the first $\lambda/2$ bits of its input. We consider the following hash function $\mathsf{H} : \tilde{\mathbb{G}}^2 \times \mathbb{Z}_q \mapsto \{0,1\}^{\lambda/2}$ based on secret key ElGamal:

- Sampling the key: sample $(\tilde{G}, k, m) \leftarrow_r \tilde{\mathbb{G}} \times \mathbb{Z}_q^2$ and set the hash key as $\tilde{\mathbf{C}} \leftarrow_r \mathsf{Enc}_{\tilde{G}}(k, m)$. Note that the key distribution is exactly the uniform distribution over $\tilde{\mathbb{G}}^2$ due to universality (see Definition 7).
- Evaluating $\mathsf{H}(\tilde{\mathbf{C}}, \cdot) : \mathsf{H}(\tilde{\mathbf{C}}, x) = \mathsf{Trunc}(\mathsf{HalfDec}(x, \tilde{\mathbf{C}}))$.

Correlation-Intractability of H. Fix a parameter $n \in \mathbb{N}$. Consider a group $\mathbb{G}$ of order $p(\lambda)$ with $\lceil \log p \rceil \approx \lambda/2n$. Fix a parameter $t \in \mathbb{Z}_p^*$ and define the set of parameters $\Lambda^t := \{(g^{\mathbf{v}}, g^{t \cdot \mathbf{v}})\}_{\mathbf{v} \in \mathbb{Z}_p^{m+1} \setminus \{\mathbf{0}\}} \subset \Lambda = (\mathbb{G}^d \setminus \{\mathbf{1}\})^2$ implicitly parameterized by the security parameter λ. Define $\mathcal{R}^{\mathsf{sparse}}_{\mathsf{LPWW}} = \{\mathcal{R}^{\mathsf{sparse}}_{\mathsf{LPWW},\mathrm{t}}\}_{t \in \mathbb{Z}_p^*}$ to be the natural sparse relation associated to the Σ-protocol of Section 4.4 for the parametrized family of languages $\mathscr{L}_{\mathsf{LPWW}}$, with repetition parameter n. That is,

$$\mathcal{R}^{\mathsf{sparse}}_{\mathsf{LPWW},\mathrm{t}} := \{(\alpha, \beta) \in \mathbb{G}^{2n} \times (\mathbb{Z}_p^*)^n : \exists x, \gamma, \mathsf{par} \in \Lambda_t \text{ s.t. } x \notin \mathscr{L}^{\mathsf{par}}_{\mathsf{LPWW}} \ \wedge \ V(x, \alpha, \beta, \gamma) = \top\},$$

where $\alpha := \{(R_i, S_i)\}_{i \in [n]}$, $\beta := \{e_i\}_{i \in [n]}$, and $\gamma := \{\mathbf{d}_i\}_{i \in [n]}$ in Figure 1. Here, the above relation can also be described alternatively using the following (inefficient) randomized function:

$$f_t(\alpha; z) : \begin{cases} \mathbb{G}^{2n} \times \mathbb{Z}_p^* \mapsto (\mathbb{Z}_p^*)^n \\ ((R_i, S_i)_{i \in [n]}, z) \to (z, ((\log_{(R_1^t / S_1)}(R_i^t / S_i)) \cdot z)_{i \in [2,n]}) \end{cases}.$$

Given this function, it is straightforward (albeit tedious) to check that the relation rewrites to

$$\mathcal{R}^{\mathsf{sparse}}_{\mathsf{LPWW},\mathrm{t}} = \{(\alpha, \beta) \in \mathbb{G}^{2n} \times (\mathbb{Z}_p^*)^n : \exists z \in \mathbb{Z}_p^*, \ f_t(\alpha; z) = \beta\}.$$

The following is the main contribution of this section.

Theorem 41. *Assume that ElGamal satisfies $2^{-\lambda/2}$-OW-KDM security with respect to efficient functions. Let $\mathcal{R}^{\mathsf{sparse}}_{\mathsf{LPWW}} = \{\mathcal{R}^{\mathsf{sparse}}_{\mathsf{LPWW},\lambda}\}_\lambda = \{\{\mathcal{R}^{\mathsf{sparse}}_{\mathsf{LPWW},\lambda,t}\}_{t \in \mathbb{Z}_p^*}\}_\lambda$ be the family of parameterized sparse relation induced by $\mathscr{L}_{\mathsf{LPWW}}$. Then the hash family $\mathcal{H} = \{\mathsf{H} : \tilde{\mathbb{G}}^2 \times \mathbb{Z}_q \mapsto \{0,1\}^{\lambda/2}\}_\lambda$ satisfies $(\varepsilon, \mathcal{R}^{\mathsf{sparse}}_{\mathsf{LPWW}})$-correlation intractability for every negligible function ε satisfying $\varepsilon(\lambda) = 2^{-o(\lambda)}$.*

Remark 42. Theorem 41 should be compared to Theorem 24 from [10]: in [10], the authors restricted their attention to a Σ-protocol with only two parallel repetitions (the language we consider is also different, but this does not matter for the conclusion – both the DDH language from [10] and the LPWW language could be used in their construction). As a consequence, they could only build a correlation-intractable hash function for their relation from the $2^{-3\lambda/4}$-OW-KDM hardness of ElGamal. By considering the general case of n parallel repetitions, and adjusting n appropriately, we

significantly strenghthen their conclusion and manage to rely on the $2^{-\lambda/2}$-OW-KDM hardness of ElGamal. By Definition 8, this means that no PPT adversary has significantly better advantage than $2^{-(1/2+o(1))\cdot\lambda}$, where the $o(1)$ in the exponent can be made smaller than $1/\lambda^{\varepsilon}$ for any constant $\varepsilon < 1$. Beyond this simple generalization, our analysis is essentially identical to that of [10]; we provide it below for the sake of completeness.

Proof. Toward proving the above theorem, we first establish the main lemma on which the proof will be based:

Lemma 43. *Let $\mathcal{A}$ be an adrversary against the $(\varepsilon, \mathcal{R}^{\mathsf{sparse}}_{\mathsf{LPWW}})$-correlation intractability of H. Then, it holds for some $t \in \mathbb{Z}_p^*$ that:*

$$\Pr_{\substack{(\tilde{G},a^*,m)\leftarrow_r \tilde{\mathbb{G}}\times\mathbb{Z}_q^2 \\ \tilde{\mathbf{C}}\leftarrow_r \mathsf{Enc}_{\tilde{G}}(a^*,m)}}[\mathcal{A}(\tilde{G},\tilde{\mathbf{C}}) = a^* | (a^*, \mathsf{H}(\tilde{\mathbf{C}}, a^*)) \in \mathcal{R}^{\mathsf{sparse}}_{\mathsf{LPWW,t}}] \geq \frac{\varepsilon(\lambda)}{2^{\frac{\lambda}{2}\cdot\left(1+\frac{1}{n}\right)}}.$$

Proof. The proof is almost identical to the proof of [10], generalized to handle an arbitrary number n of parallel repetitions. We provide it for completeness. If H is a CIH for $\mathcal{R}^{\mathsf{sparse}}_{\mathsf{LPWW}}$, then there exists $t \in \mathbb{Z}_p^*$ such that

$$\Pr\left[\begin{array}{l}(\tilde{G},k,m) \leftarrow_r \tilde{\mathbb{G}} \times \mathbb{Z}_q^2 \\ \tilde{\mathbf{C}} \leftarrow_r \mathsf{Enc}_{\tilde{G}}(k,m) \\ a \leftarrow_r \mathcal{A}(\tilde{G},\tilde{\mathbf{C}})\end{array} : (a, \mathsf{H}(\tilde{\mathbf{C}}, a)) \in \mathcal{R}^{\mathsf{sparse}}_{\mathsf{LPWW,t}}\right] \geq \varepsilon(\lambda).$$

Consider sampling independently a random input $a^* \leftarrow_r \mathbb{Z}_q$. Then we have:

$$\Pr\left[\begin{array}{l}(\tilde{G},k,m) \leftarrow_r \tilde{\mathbb{G}} \times \mathbb{Z}_q^2 \\ a^* \leftarrow_r \mathbb{Z}_q \\ \tilde{\mathbf{C}} \leftarrow_r \mathsf{Enc}_{\tilde{G}}(k,m)\end{array} : \mathcal{A}(\tilde{G},\tilde{\mathbf{C}}) = a^* \wedge (a^*, \mathsf{H}(\tilde{\mathbf{C}}, a^*)) \in \mathcal{R}^{\mathsf{sparse}}_{\mathsf{LPWW,t}}\right] \geq \frac{\varepsilon(\lambda)}{2^{\lambda}},$$

Using the (perfect) universality of ElGamal, this becomes

$$\Pr\left[\begin{array}{l}(\tilde{G},m) \leftarrow_r \tilde{\mathbb{G}} \times \mathbb{Z}_q \\ a^* \leftarrow_r \mathbb{Z}_q \\ \tilde{\mathbf{C}} \leftarrow_r \mathsf{Enc}_{\tilde{G}}(a^*,m)\end{array} : \mathcal{A}(\tilde{G},\tilde{\mathbf{C}}) = a^* \wedge (a^*, \mathsf{H}(\tilde{\mathbf{C}}, a^*)) \in \mathcal{R}^{\mathsf{sparse}}_{\mathsf{LPWW,t}}\right] \geq \frac{\varepsilon(\lambda)}{2^{\lambda}}.$$

We now introduce another (inefficient) randomized function α_t:

$$\alpha_t(\tilde{G}, a; z_1, z_2) : \begin{cases}\tilde{\mathbb{G}} \times \mathbb{Z}_q \times \{0,1\}^{\lambda/2} \times \mathbb{Z}_p^* & \mapsto \mathbb{Z}_q \\ (\tilde{G}, a; z_1, z_2) & \to \mathsf{dlog}_{\tilde{G}}(f_t(a; z_2)||z_1)\end{cases}$$

Using this function α_t, the previous inequality can be rewritten as

$$\Pr\left[\begin{array}{l}(\tilde{G},m) \leftarrow_r \tilde{\mathbb{G}} \times \mathbb{Z}_q \\ a^* \leftarrow_r \mathbb{Z}_q \\ \tilde{\mathbf{C}} \leftarrow_r \mathsf{Enc}_{\tilde{G}}(a^*,m)\end{array} : \mathcal{A}(\tilde{G},\tilde{\mathbf{C}}) = a^* \wedge \exists (z_1,z_2), m = \alpha_t(\tilde{G}, a^*; z_1, z_2)\right] \geq \frac{\varepsilon(\lambda)}{2^{\lambda}},$$

since given $\tilde{\mathbf{C}} \leftarrow_r \mathsf{Enc}_{\tilde{G}}(a^*, m)$, it holds that

$$\begin{aligned}(a^*, H(\tilde{\mathbf{C}}, a^*)) \in \mathcal{R}^{\mathsf{sparse}}_{\mathsf{LPWW,t}} &\iff \mathsf{Trunc}(\mathsf{HalfDec}(a^*, \tilde{\mathbf{C}})) = f_t(a^*; z_2) \text{ (for some } z_2) \\ &\iff \mathsf{Trunc}(\tilde{G}^m) = f_t(a^*; z_2) \\ &\iff \exists z_1, z_2, \tilde{G}^m = f_t(a^*; z_2)||z_1 \\ &\iff \exists z_1, z_2, m = \alpha_t(\tilde{G}, a^*; z_1, z_2).\end{aligned}$$

Let $S_{t,\tilde{G},a^*} := \{\alpha_t(\tilde{G}, a^*; z_1, z_2) : (z_1, z_2) \in \{0,1\}^{\lambda/2} \times \mathbb{Z}_p^*\}$ be the set of elements in $\tilde{\mathbb{G}}$ (i.e., non-truncated challenge) for which there exists an accepting last flow and word with a^* as the first flow.

Now, observe that the probability in the statement becomes:

$$\Pr\left[\begin{array}{l}(\tilde{G},a^*)\leftarrow_r \tilde{\mathbb{G}}\times\mathbb{Z}_q\\ m\leftarrow_r S_{t,\tilde{G},a^*}\\ \tilde{\mathbf{C}}\leftarrow_r \mathsf{Enc}_{\tilde{G}}(a^*,m)\end{array} : \mathcal{A}(\tilde{G},\tilde{\mathbf{C}})=a^*\right]$$

$$=\sum_{i=0}^{q-1}\Pr_{a^*\leftarrow_r\mathbb{Z}_q}[a^*=i]\cdot\Pr\left[\begin{array}{l}\tilde{G}\leftarrow_r\tilde{\mathbb{G}}\\ m\leftarrow_r S_{t,\tilde{G},i}\\ \tilde{\mathbf{C}}\leftarrow_r \mathsf{Enc}_{\tilde{G}}(i,m)\end{array} : \mathcal{A}(\tilde{G},\tilde{\mathbf{C}})=i\right]$$

$$\geq\sum_{i=0}^{q-1}\Pr_{a^*\leftarrow_r\mathbb{Z}_q}[a^*=i]\cdot\frac{\Pr\left[\begin{array}{ll}(\tilde{G},m)\leftarrow_r\tilde{\mathbb{G}}\times\mathbb{Z}_q & : \quad \mathcal{A}(\tilde{G},\tilde{\mathbf{C}})=i\\ \tilde{\mathbf{C}}\leftarrow_r\mathsf{Enc}_{\tilde{G}}(i,m) & \wedge\ (\exists z_1,z_2, m=\alpha_t(\tilde{G},i;z_1,z_2))\end{array}\right]}{\Pr[(\tilde{G},m)\leftarrow_r\tilde{\mathbb{G}}\times\mathbb{Z}_q \ : \ \exists z_1,z_2, m=\alpha_t(\tilde{G},i;z_1,z_2)]}$$

Furthermore, it holds that for any i,

$$\Pr_{m\leftarrow_r\mathbb{Z}_q}[\exists(z_1,z_2), m=\alpha_t(\tilde{G},i;z_1,z_2)]\leq 2^{-\frac{\lambda}{2}\cdot\left(1-\frac{1}{n}\right)}.$$

Indeed, the statement is equivalent to "$\exists(z_1,z_2), \tilde{G}^m=f_t(i;z_2)||z_1$", which further translates to "there exists $z_2\in\mathbb{Z}_p^*$ such that the first half of the bits of $\tilde{G}^m$ are equal to $f_t(i;z_2)$". Since a random string from $\{0,1\}^{\lambda/2}$ is equal to $f_t(i;z_2)$ with probability $2^{-\lambda/2}$ for a fixed z_2, by taking a union bound over all possible values of $z_2\in\mathbb{Z}_p^*$, where $|\mathbb{Z}_p^*|=2^{\lambda/2n}$, we get the result.

Hence, we get:

$$\Pr\left[\begin{array}{l}(\tilde{G},a^*)\leftarrow_r \tilde{\mathbb{G}}\times\mathbb{Z}_q\\ m\leftarrow_r S_{t,\tilde{G},a^*}\\ \tilde{\mathbf{C}}\leftarrow_r \mathsf{Enc}_{\tilde{G}}(a^*,m)\end{array} : \mathcal{A}(\tilde{G},\tilde{\mathbf{C}})=a^*\right]$$

$$\geq 2^{-\frac{\lambda}{2}\cdot\left(1-\frac{1}{n}\right)}\cdot\sum_{i=0}^{q-1}\Pr_{a^*\leftarrow_r\mathbb{Z}_q}[a^*=i]\cdot\Pr\left[\begin{array}{ll}(\tilde{G},m)\leftarrow_r\tilde{\mathbb{G}}\times\mathbb{Z}_q & : \quad \mathcal{A}(\tilde{G},\tilde{\mathbf{C}})=i\\ \tilde{\mathbf{C}}\leftarrow_r\mathsf{Enc}_{\tilde{G}}(i,m) & \wedge\ (\exists(z_1,z_2), m=\alpha_t(\tilde{G},i;z_1,z_2))\end{array}\right]$$

$$=2^{-\frac{\lambda}{2}\cdot\left(1-\frac{1}{n}\right)}\cdot\Pr\left[\begin{array}{ll}(\tilde{G},m,a^*)\leftarrow_r\tilde{\mathbb{G}}\times\mathbb{Z}_q^2 & : \quad \mathcal{A}(\tilde{G},\tilde{\mathbf{C}})=a^*\\ \tilde{\mathbf{C}}\leftarrow_r\mathsf{Enc}_{\tilde{G}}(a^*,m) & \wedge\ (\exists(z_1,z_2), m=\alpha_t(\tilde{G},a^*;z_1,z_2))\end{array}\right]$$

$$=\frac{\varepsilon(\lambda)}{2^{\frac{\lambda}{2}\cdot\left(1+\frac{1}{n}\right)}}.$$

This concludes the proof of the lemma.

It remains to show that this implies a contradiction to the OW-KDM security of ElGamal for *efficient* functions. The above can be rewritten as

$$\Pr_{\substack{(\tilde{G},a^*)\leftarrow_r\tilde{\mathbb{G}}\times\mathbb{Z}_q\\ m\leftarrow_r\alpha_t(\tilde{G},a^*)\\ \tilde{\mathbf{C}}\leftarrow_r\mathsf{Enc}_{\tilde{G}}(a^*,m)}}[\mathcal{A}(\tilde{G},\tilde{\mathbf{C}})=a^*]\geq\frac{\varepsilon(\lambda)}{2^{\frac{\lambda}{2}\cdot\left(1+\frac{1}{n}\right)}},\tag{1}$$

with $\alpha_t:\tilde{\mathbb{G}}\times\mathbb{Z}_q\times\{0,1\}^{\lambda/2}\times\mathbb{Z}_p^*\mapsto\mathbb{Z}_q$, such that $\alpha_t(\tilde{G},a;z_1,z_2)=\mathsf{dlog}_{\tilde{G}}(f_t(a;z_2)||z_1)$. This naturally translates to an adversary against the OW-KDM security of ElGamal where m is sampled as $\alpha_t(\tilde{G},a^*;z_1,z_2)$, but the main difficulty is that α_t is *not* an efficiently computable function. To get around this apparent issue, we use the exact same strategy as [10]. Define the (randomized) efficiently computable function f_t^{-1} as follows:

$$f_t^{-1}((e_i)_{i\in[n]};(r_i)_{i\in[n]},s_1):=\begin{cases}(\mathbb{Z}_p^*)^n\times\mathbb{Z}_p^{n+1}\mapsto\mathbb{G}^{2n}\\ ((e_i)_{i\in[n]};(r_i)_{i\in[n]},s_1)\rightarrow(g^{r_1},g^{s_1},(g^{r_i},g^{\frac{e_i(t\cdot r_1-s_1)}{e_1}-t\cdot r_i})_{i\geq 2}).\end{cases}$$

Furthermore, define F_t to be the following (efficient, randomized) function:

$$F_t(\tilde{G},m;z):\begin{cases}\tilde{\mathbb{G}}\times\mathbb{Z}_q\times\{0,1\}^{\frac{\lambda}{2}\cdot\left(1+\frac{1}{n}\right)} & \mapsto\mathbb{Z}_q\\ (\tilde{G},m;z) & \rightarrow f_t^{-1}(\mathsf{Trunc}(\tilde{G}^m);z),\end{cases}.$$

Let $\mathcal{A}$ be the previous adversary, which satisfies Equation (1). Consider the distribution of $(\tilde{G}, a^*, m)$ where we first sample $(\tilde{G}, a^*) \leftarrow_r \tilde{\mathbb{G}} \times \mathbb{Z}_p$ and set $m \leftarrow_r \alpha_t(\tilde{G}, a^*)$. Then, this is equivalent the distribution where we first sample $(\tilde{G}, m) \leftarrow_r \tilde{\mathbb{G}} \times \mathbb{Z}_q$ and then set $a^* \leftarrow_r F_t(\tilde{G}, m)$. Therefore, we have

$$\Pr_{\substack{(\tilde{G},k)\leftarrow_r\tilde{\mathbb{G}}\times\mathbb{Z}_q \\ a^*\leftarrow_r F_t(\tilde{G},k) \\ \tilde{\mathbf{C}}\leftarrow_r \mathsf{Enc}_{\tilde{G}}(a^*,k)}}[\mathcal{A}(\tilde{G}, \tilde{\mathbf{C}}) = a^*] \geq \frac{\varepsilon(\lambda)}{2^{\frac{\lambda}{2}\cdot\left(1+\frac{1}{n}\right)}}.$$

We can now build an adversary $\mathcal{B}$ against the $\mathsf{OW\text{-}KDM}$ security of ElGamal for *efficient* functions as follows: on input $(\tilde{G}, \tilde{\mathbf{C}})$, $\mathcal{B}$ parses $\tilde{\mathbf{C}}$ as $(\tilde{C}_0, \tilde{C}_1)$. $\mathcal{B}$ sets $\tilde{G}' \leftarrow \tilde{C}_0$ and $\tilde{\mathbf{C}}' \leftarrow (\tilde{G}, \tilde{C}_1)$. Then, $\mathcal{B}$ runs $\mathcal{A}(\tilde{G}', \tilde{\mathbf{C}}')$ and outputs whatever $\mathcal{A}$ outputs. Observe that the distributions

$$\{(\tilde{G}, \tilde{\mathbf{C}}) \ : \ (\tilde{G}, k) \leftarrow_r \tilde{\mathbb{G}} \times \mathbb{Z}_q, a^* \leftarrow_r F_t(\tilde{G}, k), \tilde{\mathbf{C}} \leftarrow_r \mathsf{Enc}_{\tilde{G}}(a^*, k)\},$$

which corresponds to the experiment in the previous probability, and

$$\{(\tilde{C}_0, (\tilde{G}, \tilde{C}_1)) \ : \ (\tilde{G}, k) \leftarrow_r \tilde{\mathbb{G}} \times \mathbb{Z}_q, a^* \leftarrow_r F_t(\tilde{G}, k), (\tilde{C}_0, \tilde{C}_1) \leftarrow_r \mathsf{Enc}_{\tilde{G}}(k, a^*)\}$$

are identical. Therefore,

$$\Pr_{\substack{(\tilde{G},k)\leftarrow_r\tilde{\mathbb{G}}\times\mathbb{Z}_q \\ a^*\leftarrow_r F_t(\tilde{G},k) \\ \tilde{\mathbf{C}}\leftarrow_r \mathsf{Enc}_{\tilde{G}}(k,a^*)}}[\mathcal{B}(\tilde{G}, \tilde{\mathbf{C}}) = a^*] \geq \frac{\varepsilon(\lambda)}{2^{\frac{\lambda}{2}\cdot\left(1+\frac{1}{n}\right)}},$$

which contradicts the (one-query) $2^{-(1+1/n)\lambda/2-o(\lambda)}$-hardness of $\mathsf{OW\text{-}KDM}$ security of ElGamal with respect to the family of (efficient, randomized) functions $\{F_t\}_t$ (recall that $\varepsilon(\lambda) = 2^{-o(\lambda)}$). To conclude the proof of Theorem 41, it remains to pick an appropriate value of n (note that we cannot just set $n = \lambda$ since $\lambda = 2n\lceil \log p \rceil$ depends on n). Setting e.g. $n = O(\log p)$ gives $1/n = O(1/\sqrt{\lambda}) = o(1)$, thus concluding the proof. □

IHBG-Friendly Statistical ZAP for $\mathscr{L}_{\mathsf{LPWW}}$ in Pairing-Free Groups. Equipped with the above correlation-intractable hash function, we are now ready to give our construction of our IHBG-friendly statistical ZAP. We note that this construction will actually satisfy a stronger soundness notion than required for an IHBG-friendly ZAP: adaptive computational soundness will hold for *any* parameters (an not just only for parameters sampled uniformly from $\mathsf{Col}(\mathbb{G}^d)$).[16] Let $\mathbb{G}$ be a group of order p, and let $\tilde{\mathbb{G}}$ be a group of order q such that $\lceil \log q \rceil \approx \lambda \approx 2\lceil \log p \rceil^2$. Let Π_Σ be the Σ-protocol for $\mathscr{L}_{\mathsf{LPWW}}$, with repetition parameter $n = \lceil \log p \rceil$. Let P_1, P_2 and V be the corresponding algorithms for the first and second move of the prover and the verifier, respectively. let $\mathsf{H} : \tilde{\mathbb{G}}^2 \times \mathbb{Z}_q \mapsto \{0,1\}^{\lambda/2}$ be the correlation intractable hash function constructed above.

Construction. The construction of an IHBG-friendly statistical ZAP for $\mathscr{L}_{\mathsf{LPWW}}$ with public coin length $\ell = 2\lceil \log q \rceil$, denoted as Π_{ZAP}, is described as follows.

- $\mathsf{ZAP.Prove}(\mathsf{par}, r, x, w)$: On input parameters $\mathsf{par} = (\mathbf{g}, \mathbf{h})$, a public coin $r \in \{0,1\}^\ell$, a statement $x := (X, Y) \in \mathscr{L}_{\mathsf{LPWW}}$ and a witness $w := \mathbf{y}$ such that $(X, Y) = ((\mathbf{g}^\top)^{\mathbf{y}}, (\mathbf{h}^\top)^{\mathbf{y}})$, run $\alpha \leftarrow_r P_1(\mathsf{par}, x, w)$ and compute $\beta = \mathsf{H}(r, \alpha)$, where r provides the description of the CIH hash H. Parse β as an element of $(\mathbb{Z}_p^*)^n$, and further run $\gamma \leftarrow_r P_2(\mathsf{par}, x, w, \alpha, \beta)$. Finally, output $\pi_{\mathsf{ZAP}} = (\alpha, \gamma)$.
- $\mathsf{ZAP.Verify}(\mathsf{par}, r, x, \pi_{\mathsf{ZAP}})$: On input parameters $\mathsf{par} = (\mathbf{g}, \mathbf{h})$, a public coin $r \in \{0,1\}^\ell$, a statement x, and a proof π_{ZAP}, parse π_{ZAP} as $(\alpha, \gamma) \leftarrow \pi_{\mathsf{ZAP}}$. Then, compute $\beta = \mathsf{H}(r, \alpha)$ and output $\top$ if $V(\mathsf{par}, x, \alpha, \beta, \gamma) = \top$. Otherwise, output $\bot$.

Completeness follows from the completeness of the underlying Σ-protocol.

Lemma 44. *For any parametrized language $\mathscr{L}^{\mathsf{par}}_{\mathsf{LPWW}}$ over $\mathbb{G}$, the above construction satisfies doubly-adaptive perfect witness indistinguishability.*

[16] We defined the weaker soundness notion since that was all we required to construct the IHBG protocol, and moreover, it was what we could construct from the kernel DH assumption.

Proof. As for the previous construction, this follows directly from the adaptive statistical witness indistinguishability of the underlying Σ-protocol: any attack against the witness indistinguishability of the above construction can be trivially lifted to an attack on the underlying Σ-protocol (with the same advantage), by simulating the ZAP prover by feeding the Σ-protocol prover the challenge $\beta = \mathsf{H}(r, \alpha)$, where r is any public-coin and α is the first flow from the Σ-protocol prover.

Lemma 45. *For any parametrized language $\mathscr{L}^{\mathsf{par}}_{\mathsf{LPWW}}$ over $\mathbb{G}$, the above construction satisfies adaptive computational $\varepsilon_{\mathsf{sound}}$-soundness w.r.t. colinear parameters $\mathsf{par} \in \mathsf{Col}(\mathbb{G}^d)$ under the $(\varepsilon_{\mathsf{sound}}, \mathcal{R}^{\mathsf{sparse}}_{\mathsf{LPWW}})$-correlation-intractability of H.*

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ against the adaptive computational $\varepsilon_{\mathsf{sound}}$ soundness of the above construction. Then there exists some $\mathsf{par} \in \mathsf{Col}(\mathbb{G}^d)$ such that

$$\Pr[r \leftarrow \{0,1\}^\ell, (x, \pi) \leftarrow_r \mathcal{A}(\mathsf{par}, r) : x \notin \mathscr{L}^{\mathsf{par}} \ \wedge \ \mathsf{Verify}(\mathsf{par}, r, x, \pi) = \top] \leq \epsilon_{\mathsf{sound}}.$$

Let t be such that $\mathsf{par} = (g^{\mathbf{v}}, g^{t \cdot \mathbf{v}})$. Parse $(X, Y) \leftarrow x$ and $(\alpha, \gamma) \leftarrow \pi$. Then, since $x \notin \mathscr{L}^{\mathsf{par}}_{\mathsf{LPWW}}$ and (α, β, γ) is an accepting proof, we have $(\alpha, \beta) \in \mathcal{R}^{\mathsf{sparse}}_{\mathsf{LPWW},\mathrm{t}}$. Thus, using $\mathcal{A}$, we can construct a PPT adversary $\mathcal{A}$ such that

$$\Pr_{\substack{k \leftarrow_r \tilde{\mathbb{G}}^2 \\ \alpha \leftarrow_r \mathcal{A}(k)}} [(\alpha, \mathsf{H}(k, \alpha)) \in \mathcal{R}^{\mathsf{sparse}}_{\mathsf{LPWW},\mathrm{t}}] \geq \varepsilon_{\mathsf{sound}}.$$

However, this contradicts the $(\varepsilon_{\mathsf{sound}}, \mathcal{R}^{\mathsf{sparse}}_{\mathsf{LPWW}})$-correlation intractability of our hash function.

This concludes the proof of Lemma 24. Plugging this IHBG-friendly adaptive statistical ZAP for $\mathscr{L}_{\mathsf{LPWW}}$ into the construction of IHBG of Section 5 and combining it with the construction of statistical ZAP for NP from any IHBG from Section 3, we get our second main theorem:

Theorem 46 (Statistical ZAPs in Pairing-Free Groups). *Assume that the explicit μ-hardness of the DDH assumption holds in a group $\mathbb{G}$ of order p for any negligible function μ (which can be arbitrarily close to an inverse polynomial function), and that the $2^{-\lambda/2}$-OW-KDM security of ElGamal holds over a group $\tilde{\mathbb{G}}$ of order q such that $\lceil \log q \rceil \approx \lambda \approx 2\lceil \log p \rceil^2$. Then there exists an adaptive statistically witness indistinguishable ZAP for NP with non-adaptive computational soundness.*

As we mentioned in the preliminaries, the above theorem can be easily generalized to hold under any of the more general family of Diffie-Hellman assumption, such as the (decisional and kernel) matrix Diffie-Hellman assumption [15, 38] (which includes for example the k-Lin assumption from [27]).

References

1. Badrinarayanan, S., Fernando, R., Jain, A., Khurana, D., Sahai, A.: Statistical ZAP arguments. In: Rijmen, V., Ishai, Y. (eds.) EUROCRYPT 2020, Part III. pp. 642–667. LNCS, Springer, Heidelberg (May 2020)
2. Badrinarayanan, S., Garg, S., Ishai, Y., Sahai, A., Wadia, A.: Two-message witness indistinguishability and secure computation in the plain model from new assumptions. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part III. LNCS, vol. 10626, pp. 275–303. Springer, Heidelberg (Dec 2017)
3. Bitansky, N., Paneth, O.: ZAPs and non-interactive witness indistinguishability from indistinguishability obfuscation. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 401–427. Springer, Heidelberg (Mar 2015)
4. Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications (extended abstract). In: 20th ACM STOC. pp. 103–112. ACM Press (May 1988)
5. Canetti, R., Chen, Y., Holmgren, J., Lombardi, A., Rothblum, G.N., Rothblum, R.D., Wichs, D.: Fiat-Shamir: from practice to theory. In: Charikar, M., Cohen, E. (eds.) 51st ACM STOC. pp. 1082–1090. ACM Press (Jun 2019)
6. Canetti, R., Chen, Y., Reyzin, L., Rothblum, R.D.: Fiat-Shamir and correlation intractability from strong KDM-secure encryption. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part I. LNCS, vol. 10820, pp. 91–122. Springer, Heidelberg (Apr / May 2018)
7. Cash, D., Kiltz, E., Shoup, V.: The twin Diffie-Hellman problem and applications. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 127–145. Springer, Heidelberg (Apr 2008)

8. Couteau, G., Hartmann, D.: Shorter non-interactive zero-knowledge arguments and ZAPs for algebraic languages. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2020, Part III. pp. 768–798. LNCS, Springer, Heidelberg (Aug 2020)
9. Couteau, G., Hofheinz, D.: Designated-verifier pseudorandom generators, and their applications. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part II. LNCS, vol. 11477, pp. 562–592. Springer, Heidelberg (May 2019)
10. Couteau, G., Katsumata, S., Ursu, B.: Non-interactive zero-knowledge in pairing-free groups from weaker assumptions. In: Rijmen, V., Ishai, Y. (eds.) EUROCRYPT 2020, Part III. pp. 442–471. LNCS, Springer, Heidelberg (May 2020)
11. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk, H. (ed.) CRYPTO'98. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (Aug 1998)
12. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (Apr / May 2002)
13. Dwork, C., Naor, M.: Zaps and their applications. In: 41st FOCS. pp. 283–293. IEEE Computer Society Press (Nov 2000)
14. Dwork, C., Naor, M.: Zaps and their applications. SIAM J. Comput. 36(6), 1513–1543 (2007)
15. Escala, A., Herold, G., Kiltz, E., Råfols, C., Villar, J.: An algebraic framework for Diffie-Hellman assumptions. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 129–147. Springer, Heidelberg (Aug 2013)
16. Feige, U., Lapidot, D., Shamir, A.: Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In: 31st FOCS. pp. 308–317. IEEE Computer Society Press (Oct 1990)
17. Feige, U., Lapidot, D., Shamir, A.: Multiple noninteractive zero knowledge proofs under general assumptions. SIAM J. Comput. 29(1), 1–28 (1999)
18. Feige, U., Shamir, A.: Witness indistinguishable and witness hiding protocols. In: 22nd ACM STOC. pp. 416–426. ACM Press (May 1990)
19. Gentry, C., Wichs, D.: Separating succinct non-interactive arguments from all falsifiable assumptions. In: Fortnow, L., Vadhan, S.P. (eds.) 43rd ACM STOC. pp. 99–108. ACM Press (Jun 2011)
20. Goldreich, O., Oren, Y.: Definitions and properties of zero-knowledge proof systems. Journal of Cryptology 7(1), 1–32 (Dec 1994)
21. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM J. Comput. 18(1), 186–208 (1989)
22. Goyal, V., Jain, A., Jin, Z., Malavolta, G.: Statistical zaps and new oblivious transfer protocols. In: Rijmen, V., Ishai, Y. (eds.) EUROCRYPT 2020, Part III. pp. 668–699. LNCS, Springer, Heidelberg (May 2020)
23. Groth, J., Ostrovsky, R., Sahai, A.: Non-interactive zaps and new techniques for NIZK. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 97–111. Springer, Heidelberg (Aug 2006)
24. Groth, J., Ostrovsky, R., Sahai, A.: Perfect non-interactive zero knowledge for NP. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 339–358. Springer, Heidelberg (May / Jun 2006)
25. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (Apr 2008)
26. Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM Journal on Computing 28(4), 1364–1396 (1999)
27. Hofheinz, D., Kiltz, E.: Secure hybrid encryption from weakened key encapsulation. Cryptology ePrint Archive, Report 2007/288 (2007), `http://eprint.iacr.org/2007/288`
28. Jain, A., Kalai, Y.T., Khurana, D., Rothblum, R.: Distinguisher-dependent simulation in two rounds and its applications. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part II. LNCS, vol. 10402, pp. 158–189. Springer, Heidelberg (Aug 2017)
29. Jain, A., Zhengzhong, J.: Non-interactive zero knowledge from sub-exponential ddh. In: EUROCRYPT 2021 (2021)
30. Kalai, Y.T., Khurana, D., Sahai, A.: Statistical witness indistinguishability (and more) in two messages. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part III. LNCS, vol. 10822, pp. 34–65. Springer, Heidelberg (Apr / May 2018)
31. Katsumata, S., Nishimaki, R., Yamada, S., Yamakawa, T.: Designated verifier/prover and preprocessing NIZKs from Diffie-Hellman assumptions. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part II. LNCS, vol. 11477, pp. 622–651. Springer, Heidelberg (May 2019)
32. Khurana, D., Sahai, A.: How to achieve non-malleability in one or two rounds. In: Umans, C. (ed.) 58th FOCS. pp. 564–575. IEEE Computer Society Press (Oct 2017)
33. Kiltz, E., Wee, H.: Quasi-adaptive NIZK for linear subspaces revisited. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part II. LNCS, vol. 9057, pp. 101–128. Springer, Heidelberg (Apr 2015)

34. Libert, B., Passelègue, A., Wee, H., Wu, D.J.: New constructions of statistical NIZKs: Dual-mode DV-NIZKs and more. In: Rijmen, V., Ishai, Y. (eds.) EUROCRYPT 2020, Part III. pp. 410–441. LNCS, Springer, Heidelberg (May 2020)
35. Lindell, Y.: An efficient transform from sigma protocols to NIZK with a CRS and non-programmable random oracle. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part I. LNCS, vol. 9014, pp. 93–109. Springer, Heidelberg (Mar 2015)
36. Lombardi, A., Vaikuntanathan, V., Wichs, D.: 2-message publicly verifiable WI from (subexponential) LWE. Cryptology ePrint Archive, Report 2019/808 (2019), https://eprint.iacr.org/2019/808
37. Lombardi, A., Vaikuntanathan, V., Wichs, D.: Statistical ZAPR arguments from bilinear maps. In: Rijmen, V., Ishai, Y. (eds.) EUROCRYPT 2020, Part III. pp. 620–641. LNCS, Springer, Heidelberg (May 2020)
38. Morillo, P., Ràfols, C., Villar, J.L.: Matrix computational assumptions in multilinear groups. Cryptology ePrint Archive, Report 2015/353 (2015), http://eprint.iacr.org/2015/353
39. Naor, M.: On cryptographic assumptions and challenges (invited talk). In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 96–109. Springer, Heidelberg (Aug 2003)
40. Pass, R.: Unprovable security of perfect NIZK and non-interactive non-malleable commitments. In: Sahai, A. (ed.) TCC 2013. LNCS, vol. 7785, pp. 334–354. Springer, Heidelberg (Mar 2013)
41. Quach, W., Rothblum, R.D., Wichs, D.: Reusable designated-verifier NIZKs for all NP from CDH. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part II. LNCS, vol. 11477, pp. 593–621. Springer, Heidelberg (May 2019)

Table of Contents