Somewhere Statistical Soundness, Post-Quantum Security, and SNARGs

Yael Tauman Kalai* Microsoft Research

Vinod Vaikuntanathan† MIT

Rachel Yun Zhang‡ MIT

July 20, 2021

Abstract

We instantiate Kilian's protocol with a computationally non-signaling PCP (Brakerski, Holmgren, and Kalai, STOC 2017) and a somewhere statistically binding hash family (Hubacek and Wichs, ITCS 2015). Observing that the first two messages of Kilian's protocol, instantiated with these primitives, is a sound instantiation of the BMW heuristic (Kalai, Raz, and Rothblum, STOC 2013), we show how to efficiently convert any succinct non-interactive argument (SNARG) for BatchNP into a SNARG for any language that has a non-signaling PCP, including any deterministic language and any language in NTISP, using a somewhere statistically binding hash family.

We also introduce the notion of a somewhere statistically sound (SSS) interactive argument, which is a hybrid between a statistically sound proof and a computationally sound proof (a.k.a. an argument).

- We show that Kilian's protocol, instantiated in the above way, is an SSS argument.
- Secondly, we show that the soundness of SSS arguments can be proved in a straight-line manner, implying that they are also post-quantum sound if the underlying assumption is post-quantum secure. This provides a straightforward proof that Kilian's protocol, instantiated as above, is post-quantum sound under the post-quantum hardness of LWE (though we emphasize that a computationally non-signaling PCP is known to exist only for deterministic languages and for specific subclasses of non-deterministic languages such as NTISP, but not for all of NP).
- We put forward a natural conjecture that constant-round SSS arguments can be soundly converted into non-interactive arguments via the Fiat-Shamir transformation. We argue that SSS arguments evade the current Fiat-Shamir counterexamples, including the one for Kilian's protocol (Bartusek, Bronfman, Holmgren, Ma and Rothblum, TCC 2019) by requiring additional properties from both the hash family and the PCP.

*E-mail:yael@microsoft.com

†E-mail:vinodv@csail.mit.edu

‡E-mail:rachelyz@mit.edu

Contents

1 Introduction

In the past decade, there has been a significant effort to construct efficiently verifiable, succinct, and non-interactive argument systems (also called SNARGs).[1] One way of constructing such argument systems is by first constructing a *public-coin* interactive proof or argument system, and then eliminating the interaction using the celebrated Fiat-Shamir paradigm [FS86].

The Fiat-Shamir paradigm provides a generic way of converting any public-coin interactive protocol into a non-interactive one, in the common random string (CRS) model. Loosely speaking, the Fiat-Shamir paradigm converts an interactive proof $(\mathcal{P}, \mathcal{V})$ for a language L to a non-interactive argument $(\mathcal{P}', \mathcal{V}')$ for L in the CRS model. The CRS consists of randomly chosen hash functions $h_1, \ldots, h_\ell$ from a hash family $\mathcal{H}$, where ℓ is the number of rounds in the protocol $(\mathcal{P}, \mathcal{V})$. To compute a non-interactive proof for $x \in L$, the non-interactive prover $\mathcal{P}'(x)$ generates a transcript corresponding to $(\mathcal{P}, \mathcal{V})(x)$, denoted by $(\alpha_1, \beta_1, \ldots, \alpha_\ell, \beta_\ell)$, by emulating $\mathcal{P}(x)$ and replacing each verifier message β_i by $\beta_i = h_i(\alpha_1, \beta_1, \ldots, \alpha_{i-1}, \beta_{i-1}, \alpha_i)$. The verifier $\mathcal{V}'(x)$ accepts if and only if $\mathcal{V}(x)$ accepts this transcript and $\beta_i = h_i(\alpha_1, \beta_1, \ldots, \alpha_{i-1}, \beta_{i-1}, \alpha_i)$ for every $i \in [\ell]$.

This paradigm has been extremely influential in practice, and its soundness has been extensively studied. For statistically sound proofs, this paradigm is believed to be sound, at least under strong computational assumptions [KRR17, CCRR18, HL18, CCH+19]. Moreover, for some protocols such as the Goldwasser-Kalai-Rothblum protocol [GKR08] and several zero-knowledge protocols for NP such as Blum's Hamiltonicity protocol [Blu86] and the GMW 3-coloring protocol [GMW91], this paradigm is provably sound under the polynomial or sub-exponential hardness of learning with errors (LWE) [CCH+19, PS19, JKKZ21, HLR21], which are standard assumptions.

On the other hand, for computationally sound proofs (known as arguments) the situation is quite grim. There are (contrived) examples of interactive arguments for which the resulting non-interactive argument obtained by applying the Fiat-Shamir paradigm is not sound, no matter which hash family is used [Bar01a, GK05]. Moroever, recently it was shown that the Fiat-Shamir paradigm is not sound when applied to the celebrated Kilian's protocol [BBH+19]. This begs the following question:

> *Does there exist an interesting class of interactive arguments for which the Fiat-Shamir paradigm* **is** *sound?*

1.1 Somewhere Statistically Sound Interactive Arguments

We introduce a family of interactive arguments that we call *somewhere statistically sound* (SSS). In what follows, we always assume (w.l.o.g.) that the first message in the protocol is sent by the verifier. An interactive argument $(\mathcal{P}, \mathcal{V})$ is said to be SSS if for every legal first message β_1, there exists a third message $\beta_2 = T(\beta_1)$ sent by the verifier such that the following two properties hold:

- For every PPT (cheating) prover $\mathcal{P}^*$, conditioned on the first three messages being $(\beta_1, \mathcal{P}^*(\beta_1), T(\beta_1))$ the remaining protocol is statistically sound with overwhelming probability over β_1. Namely, for any $x \notin L$ and any PPT $\mathcal{P}^*$, with overwhelming probability, any

[1] An argument system is a computationally sound proof system.

(even all powerful) cheating prover cannot convince the verifier to accept $x \notin L$ except with negligible probability, *conditioned on the first three messages being* $(\beta_1, \mathcal{P}^*(\beta_1), T(\beta_1))$.

- The pair $(\beta_1, T(\beta_1))$ is computationally indistinguishable from a random pair (β_1, β_2) of the verifier's first two messages. We emphasize that this in particular implies that the function T has to be computationally inefficient.

As we argue below, SSS interactive arguments are of great interest for several reasons.

1. First, we prove that such protocols are post-quantum sound, if the assumption that they rely on is post-quantum secure. We note that in general, interactive protocols that are proven classically secure under post-quantum assumptions are *not* post-quantum secure. This is because the proof of security often relies on the rewinding technique, which is not generally applicable in the quantum setting due to the fact that quantum states are not clonable [Wat09, Unr12]. We show that SSS arguments have a *straight-line* proof of soundness (i.e., without rewinding the cheating prover), and are thus immediately post-quantum sound. We elaborate on this in Section 1.1.1.

2. Second, we prove that Kilian's protocol, instantiated with (a repeated version of) a somewhere-statistically-binding (SSB) hash family [HW15] (for which constructions based on the LWE assumption exist) and a PCP with computational non-signaling soundness, is SSS. We elaborate on this in Section 1.2. Combined with (1), this provides a rather simple proof of post-quantum soundness of Kilian's protocol, comprehensible to a "quantum dummy."[2]

   We note that this instantiation of Kilian's protocol is only for deterministic languages and for specific classes of non-deterministic languages (such as BatchNP or the class $\mathsf{NTISP}(t, s)$[3]) since a computationally non-signaling PCP with the desired parameters exists only for such languages (where for languages in BatchNP the communication complexity grows with the length of a single witness and for $\mathsf{NTISP}(t, s)$ the communication complexity grows with the space s). Proving that the classical Kilian protocol [Kil92] is post-quantum sound for all of NP was a grand challenge, and was only very recently resolved by Chiesa, Ma, Spooner and Zhandry [CMSZ21] using highly non-trivial quantum techniques.

3. Finally, we *conjecture* that any SSS interactive argument is *Fiat-Shamir friendly*; meaning that for any SSS interactive argument $(\mathcal{P}, \mathcal{V})$ there exists a hash family $\mathcal{H}$ such that applying the Fiat-Shamir paradigm w.r.t. $\mathcal{H}$ to $(\mathcal{P}, \mathcal{V})$ results with a sound non-interactive argument. Indeed, prior to this work, the only interactive argument that was proven to be Fiat-Shamir friendly, in the work of Canetti et al [CSW20], is an SSS argument, and was used to construct a (non-succint) UC NIZK for NP with adaptive soundness guarantees.

   We emphasize that we do not prove that any SSS interactive argument is Fiat-Shamir friendy, only conjecture it. We believe that it is a promising path for obtaining a SNARG for all

[2] https://simons.berkeley.edu/events/quantum-lectures-crypto-dummies

[3] The class $\mathsf{NTISP}(t, s)$ consists of all the languages that are decidable by a non-deterministic Turing machine running in time t and space s.

deterministic languages based on a standard post-quantum secure assumption. In particular, we conjecture that Kilian's protocol, with the instantiations above (SSB hash family and PCP with computational non-signaling soundness) is Fiat-Shamir friendly.

This is in contrast with the recent work [BBH+19] that showed that in general, Kilian's protocol is *not* Fiat-Shamir friendly. We remark that it was already suggested in [BBH+19] to use an SSB hash family as one step to evade their impossibility result. We suggest that the Fiat-Shamir transform indeed works using (repeated) SSB hashing *combined with* a (computationally) non-signaling PCP, giving us succinct non-interactive arguments for all of P (and even more, as described above). We leave this as an important open problem, and believe it worth significant cycles from the community.

An additional conceptual contribution of this work is our view of the first two messages of Kilian's protocol, using a (repeated) SSB hash function and a computational non-signaling PCP, as an instantiation of the 2-message protocol of [KRR13, BHK17] (itself a provable version of an idea originally due to Biehl, Meyer and Wetzel [BMW99]). Using this view, we show how to construct a SNARG for any language that has a computational non-signaling PCP (including any language in DTIME(t) [KRR14, BHK17] and any language in NTISP(t, s) [BKK+18]) from any SNARG for BatchNP. We elaborate on this in Section 1.3.

Concurrent Work. In a concurrent and independent work, Choudhuri, Jain and Jin [CJJ21] construct SNARGs for BatchNP from LWE. Thus, using their result we obtain a SNARG for any language that has a computational non-signaling PCP from the LWE assumption. In particular, as we elaborate on in Section 1.3, we obtain a SNARG for any language in DTIME(t) or in NTISP(t, s) with communication complexity $\mathrm{polylog}(t)$ or $\mathrm{polylog}(s, \log t)$, respectively, from the sub-exponential hardness of LWE. We note that [CJJ21] also showed how to use their SNARG for BatchNP to construct a SNARG for P as well as for RAM computations.

1.1.1 SSS and Straight-Line Soundness

In a nutshell, the reason that any SSS protocol is post-quantum sound is due to the fact that it has *straight-line soundness*, meaning that any (even quantum) successful cheating prover can be used in a black box and straight-line manner (without rewinding) to break some complexity assumption.

Theorem 1.1 (Informal)**.** *Any* SSS *interactive argument has a* straight-line *soundness proof.*

Loosely speaking, we prove this theorem as follows. Fix any SSS interactive argument $(\mathcal{P}, \mathcal{V})$ for a language $\mathcal{L}$. We construct a (uniform) PPT black-box reduction $\mathcal{R}$, that takes as input a pair (β_1, β_2), and distinguishes between the case that $\beta_2 = T(\beta_1)$ and the case that β_2 is chosen at random, given black-box and staight-line access to any (even quantum) cheating prover $\mathcal{P}^*$.

The reduction $\mathcal{R}$ works as follows: It runs the cheating prover with β_1, and then upon recieving $\alpha_1 = \mathcal{P}^*(\beta_1)$, it sends $\mathcal{P}^*$ the challenge β_2. The reduction then continues emulating the honest verifier until the end of the protocol. If the transcript is accepting, then $\mathcal{R}$ outputs 1 (indicating that β_2 is random), and otherwise it outputs 0. By the assumption that $\mathcal{P}^*$ is convincing with

non-negligible probability, if β_1 and β_2 are random then the transcript is accepting with non-negligible probability. On the other hand, by the SSS property, if $\beta_2 = T(\beta_1)$, then the transcript is accepted with only negligible probability. Thus, the reduction $\mathcal{R}$ outputs 1 with probability that is non-negligibly larger in the case that β_2 is random, as desired.

We note that any interactive argument that has a straight-line soundness proof is immediately post-quantum sound, assuming that the underlying assumption is post-quantum secure. This is the case since the analysis above extends readily to the quantum setting. As mentioned above, this is in contrast to the standard analysis which uses rewinding, and hence often fails in the post-quantum setting.

Claim 1.2 (Informal)**.** *Any* SSS *interactive argument where both* SSS *properties are straight-line reducible from an assumption* A *is also post-quantum sound if assumption* A *holds w.r.t. quantum adversaries.*

This property makes SSS arguments particularly appealing, given the major effort by the community to make cryptographic protocols post-quantum secure.

1.1.2 SSS and Fiat-Shamir Friendliness

Another reason why SSS arguments are of interest is that we believe (and conjecture) that such protocols are "Fiat-Shamir friendly."

Conjecture 1.3. *Any constant round* SSS *interactive argument* $(\mathcal{P}, \mathcal{V})$ *is* Fiat-Shamir friendly*.*

Recall that we believe that any (constant round) statistically sound proof is Fiat-Shamir friendly, whereas we know that this is false for computationally sound proofs. Thus, it is natural to ask whether the hybrid class of all (constant round) SSS interactive arguments is Fiat-Shamir friendly.

We note that all known negative results for the Fiat-Shamir paradigm [Bar01b, GK03, CMSZ21] are for arguments that are *not* SSS. In particular, these interactive arguments are constructed by adding an additional accepting clause, such that if the prover can predict the verifier's next message then he can easily convince the verifier to accept this alternative clause (even false statements). This does not harm soundness in the interactive setting since the interactive prover cannot predict the verifier's next message and hence cannot use this additional clause. On the other hand, when Fiat-Shamir is applied, the prover can, by definition, use the description of the hash function to predict the verifier's next message, harming the soundness of the non-interactive protocol and thus demonstrating the insecurity of the Fiat-Shamir paradigm.

Crucially, we emphasize that this additional clause makes the resulting argument *not* SSS, since this additional clause inherently does not have statistical soundness. This is the case because the witness for this additional clause (which is the Fiat-Shamir hash function) can be larger than the communication complexity, and hence to verify this clause we must use a *succinct* argument. Importantly, we note that even if this clause is SSS the entire protocol is not, since this clause is executed after the first two messages.

We note that Bartusek et al. [BBH+19] give an instantiation of Kilian's protocol for the trivial (empty) language for which applying the Fiat-Shamir paradigm provably results in a sound

protocol. Their instantiation employs an SSB hash function and a particular PCP for the empty language, and the protocol is in fact SSS. Indeed, our conjecture is a stronger statement, namely that the notion of somewhere statistical soundness is sufficient to apply Fiat-Shamir soundly.

1.2 Instantiating an SSS version of Kilian

We show an instantiation of Kilian's protocol which is SSS. Specifically, we prove that if we use a PCP that has computational non-signaling soundness, and denote its query complexity (or more precisely, its locality) by ℓ, and if we tree-commit to this PCP ℓ times using ℓ somewhere statistically binding (SSB) hash functions [HW15], then the resulting protocol is SSS. In particular, we obtain the following corollary.

Theorem 1.4 (Informal). *Kilian's protocol is* SSS*, and thus has post-quantum soundness, if we use a* PCP *with computational non-signaling soundness with locality ℓ, and if the prover tree-commits to this* PCP *using ℓ post-quantum* SSB *hash functions.*

Hubáček and Wichs [HW15] constructed an SSB hash family assuming the hardness of LWE. This hash family is post-quantum secure assuming the post-quantum hardness of LWE. Moreover, [KRR14, BHK17] constructed a PCP with computational non-signaling soundness for all deterministic languages and BatchNP languages, where for languages in $\mathsf{DTIME}(t)$ the query complexity (or more precisely, the locality) is $\mathrm{polylog}(t)$, and for BatchNP languages it is $m \cdot \mathrm{polylog}(N)$, where m is the length of a single witness and N is the number of instances in the batch. These two results, together with Theorem 1.4, imply the following corollary.

Corollary 1.5 (Informal). *There exists an instantiation of Kilian's protocol that is* SSS*, and thus post-quantum sound, for all deterministic computations and* BatchNP *languages, assuming the sub-exponential post-quantum hardness of* LWE*. For* $\mathsf{DTIME}(t)$ *languages the communication complexity grows with* $\mathrm{polylog}(t)$ *and for* BatchNP *languages the communicaiton complexity grows with $m \cdot \mathrm{polylog}(N)$, where m is the length of a single witness and N is the number of instances in the batch.*

More generally, we prove that if the PCP is Ω-computational non-signaling, with locality ℓ, and if we tree-commit to this PCP using ℓ hash functions that are SSB with Ω-security (i.e., even a $\mathrm{poly}(\Omega)$-size adversary cannot break the SSB with probability non-negligible in Ω), then the resulting Kilian's protocol is SSS. We note that [BKK+18] constructed a PCP with 2^s-computational non-signaling soundness (with locality $\mathrm{poly}(s)$) for any language in $\mathsf{NTISP}(t, s)$ with $s \geq \log t$. In addition, the SSB from [HW15] is 2^s-secure (where the key size grows with s) assuming the sub-exponential hardness of LWE. We thus obtain the following corollary.

Corollary 1.6 (Informal). *For $s \geq \log t$, there exists an instantiation of Kilian's protocol for all languages in* $\mathsf{NTISP}(t, s)$ *that is* SSS*, and thus post-quantum sound, where the communication complexity grows with s, assuming the sub-exponential post-quantum hardness of* LWE*.*

As mentioned above, we conjecture that this instantiation is Fiat-Shamir friendly, and leave the proof (or refutation) of this conjecture as an important open problem.

1.3 From BMW/KRR Back to Kilian

We take a somewhat anachronistic view and see Kilian's *four-message, public-coin* interactive argument as a natural interpolation of the Biehl-Meyer-Wetzel (BMW) heuristic. Recall that the BMW heuristic takes any PCP and any single-sever PIR scheme, and uses them to construct a two-message (succinct) argument where each PCP query is sent to the prover as a PIR query (see Section 2.5 for more details). The BMW heuristic is not known to be sound in general [DLN+04, DHRW16]; however, it is known to be computationally sound if the PCP is computational non-signaling [KRR13, BHK17]. The protocol is privately verifiable since the verifier needs to run the PIR decoding on the prover's message, in a sense decrypting it.

To see the relation to Kilian's protocol with an SSB hash and a non-signaling PCP, recall that an SSB hash family is a hash family $\mathcal{H}$ where each hash seed s is associated with an index $i \in [N]$, where N is the length of the input, such that $h_s(x)$ is statistically binding on x_i, and one can extract x_i from the hash value $h_s(x)$ given a trapdoor t that is generated together with s (see Section 2.3, Definition 2.11). In our instantiation of Kilian's protocol, we hash the PCP with ℓ SSB hash functions, where ℓ is the locality parameter of the PCP. Namely, the verifier's first message is $s_1, \ldots, s_\ell$ and the prover's response is $(h_{s_1}(\pi), \ldots, h_{s_\ell}(\pi))$. By the semantic security property of a SSB hash family, and its inversion property given the trapdoor, this hash function can be thought of as a PIR scheme. Thus, these first two messages of Kilian's protocol are nothing but an instantiation of the BMW heuristic.

We know that this heuristic is not sound in general [DLN+04, DHRW16], yet it is sound if the underlying PCP has non-signaling soundness [KRR14, BHK17]. Indeed, until very recently, almost all two-message succinct arguments that were proven sound under standard assumptions relied on this heuristic (and used non-signaling PCPs). The main downside of this approach is that it yields a *privately verifiable* (a.k.a. designated verifier) argument. Converting this protocol to a publicly verifiable one is a major open problem. One attempt was made in [KPY19], which converted it to being publicly verifiable by relying on *zero-testable encryption scheme*, and a construction of this cryptographic primitive was given under a complexity assumption on groups with bilinear maps. This left open the problem of relying on more standard and ideally post-quantum secure assumptions.

Our instantiation of Kilian's protocol can be thought of as a way of converting the BMW protocol to a publicly verifiable one, albeit at a cost of adding two rounds. In this instantiation, we execute the BMW heuristic, but *the verifier never decrypts the* PIR *answer*. Instead, we view the PIR answer as a commitment to the PCP, and we add two messages, where the verifier sends PCP queries in the clear, and *the prover decommits to the answers*. These additional messages are in lieu of the verifier decrypting the PIR answers by himself.

In summary, consider the goal of constructing a *post-quantum-secure publicly verifiable* SNARG for all of P. One could either start with a privately verifiable SNARG for all of P, namely the KRR protocol whose security relies on the LWE assumption, and try to make it publicly verifiable using, e.g., techniques from [KPY19]. However, it is currently unclear how to apply these techniques outside of the bilinear maps world. One alternate path is to first do *round-expansion* of KRR into a Kilian-like protocol (while instantiating the CRHF with an SSB hash family, and instantiating the PCP with a computational non-signaling PCP), and then *round-reduce* it using

Fiat-Shamir. In the next section, we suggest yet another path, by using a SNARG for BatchNP.

1.4 SNARGs: from BatchNP to P and Beyond

This view of the first two messages of the Kilian protocol as an instantiation of KRR, leads us to our final contribution, which is an alternative pathway to getting a SNARG for all of P, or more generally, for any language that has a computational non-signaling PCP, including $\mathsf{DTIME}(t)$ and $\mathsf{NTISP}(t, s)$. We show a reduction from constructing succinct non-interactive arguments (SNARGs) for this class of languages into the potentially simpler goal of constructing SNARGs for BatchNP.

The starting point is the two-round preamble where the verifier sends the prover the descriptions of ℓ SSB hash functions, and the verifier replies with ℓ Merkle roots of a non-signaling PCP. The key observation is that the remainder of the protocol can be a proof of the following BatchNP statement (which can be communicated in the first two rounds as well): for every possible query set Q generated by the PCP verifier, there are values of π_Q as well as openings o_Q such that (a) (π_Q, o_Q) constitutes valid Merkle paths to the ℓ roots; and (b) the PCP verifier accepts (Q, π_Q).

We argue that this 2-message protocol is sound: if the instance being proven is false, then by the soundness of KRR the answers that are committed to by the Merkle roots are rejecting, and hence by the somewhere-statistical binding property, the resulting BatchNP statement is false. Therefore, it seems that all we need to instantiate this approach is a SNARG for BatchNP.

There are several issues that come up in making this idea work. First, if the PCP has negligible soundness error, then the number of possible query sets generated by the verifier is superpolynomially large, meaning that the (honest) prover runtime is superpolynomial. Fortunately, all known PCP constructions (including the one from [KRR14, BKK+18]) have the property that each query set can be partitioned into a set of "tests," where the queries in each test and their corresponding answers can be verified on their own, and importantly, the number of possible tests is polynomial.[4] Therefore, our BatchNP statement should rather be that for every test ζ there are values of π_ζ as well as openings o_ζ such that (a) $(\pi_\zeta, \mathsf{o}_\zeta)$ constitutes a valid Merkle path; and (b) the PCP verifier accepts (ζ, π_ζ). Note that this BatchNP statement is polynomially large.

Secondly, even though we ensured that the number of instances in the BatchNP statement is polynomial, this polynomial, denoted by N, is at least as large as the runtime of the underlying computation. Note that even though the proof length scales only poly-logarithmically with N, the *verifier runtime* scales at least linearly with N since the verifier needs to at least read the entire statement. To solve this, we observe that in our case, the BatchNP statement actually has a succinct description. Thus, if there are succinct, easy to verify, proofs for succinctly specified BatchNP statements, we are back in business. We note that even if this is not the case, if the verifier's verdict function can be computed by a circuit that has depth only $\mathrm{polylog}(N)$ (but size $\mathrm{poly}(N)$), then again we are in business since we can use the SNARG for bounded depth computations (from sub-exponential LWE) [JKKZ20], and delegate this computation back to the prover.

Third and finally, note that the BatchNP proof system must have adaptive soundness since the prover gets to choose the BatchNP statement, in particular the Merkle roots, *after* he recieves the

[4] For example, the tests in the PCP of [BFLS91] (and in the PCP of [KRR14, BKK+18]) are either low-degree tests or consistency tests.

CRS/first message of the BatchNP proof. Since the Merkle roots are small in size, this can be easily handled by complexity leveraging. We therefore only require non-adaptive soundness with appropriate security. We elaborate on this in Section 5.

2 Preliminaries

2.1 Straight-Line Reductions

In this section, we define the notion of straight-line soundness, and more generally straight-line reductions.

Definition 2.1. *(Straight-Line Reductions) We say that an interactive argument $(\mathcal{P}, \mathcal{V})(1^\kappa)$ for a language $\mathcal{L} = \{\mathcal{L}_n\}_{n\in\mathbb{N}}$ is $\theta = \theta(\kappa)$-*straight-line sound *if there is a* PPT *black box reduction $\mathcal{R}$ and a non-interactive θ-decisional complexity assumption* [GK16],[5] *such that $\mathcal{R}$, given oracle access to any cheating prover $\mathcal{P}^*$ that breaks soundness with probability $1/\mathrm{poly}(\theta)$, interacts with $\mathcal{P}^*$* once *(without rewinding) by sending $\mathcal{P}^*$ a single message for each round, and using the transcript obtained, breaks the assumption.*

*More generally, we say that a primitive is $\theta = \theta(\kappa)$-*straight-line secure *(or θ-secure via a straight-line reduction, or its security proof is θ-straight line) if there is a* PPT *black box reduction $\mathcal{R}$ and a non-interactive θ-decisional complexity assumption*[6] *such that $\mathcal{R}$, given oracle access to any size-$\mathrm{poly}(\theta)$ adversary $\mathcal{A}$ that breaks the security of the primitive with probability $1/\mathrm{poly}(\theta)$, interacts with $\mathcal{A}$* once *(without rewinding) and, using the transcript obtained, breaks the assumption.*

Definition 2.2 ([GK16])**.** *An assumption is a θ-decisional complexity assumption if it is associated with two probabilistic polynomial-time distributions $(\mathcal{D}_0, \mathcal{D}_1)$, such that for any $\mathrm{poly}(\theta)$-size algorithm $\mathcal{A}$ there exists a negligible function μ such that for any $\kappa \in \mathbb{N}$,*

$$\left| \Pr_{x\leftarrow\mathcal{D}_0(1^\kappa)}[\mathcal{A}(x) = 1] - \Pr_{x\leftarrow\mathcal{D}_1(1^\kappa)}[\mathcal{A}(x) = 1] \right| \leq \mu(\theta(\kappa)).$$

2.2 Probabilistically Checkable Proofs

We recall the definition of a *probabilistically checkable proof* (PCP) and that of a *non-signaling* PCP. Intuitively, a PCP is a function Π that takes as input a proof (for example, a witness w for an NP statement x), and converts it into another proof $\pi = \Pi(x, w)$ which can be verified by a randomized verifier that reads only a few of its bits.

Definition 2.3 (PCP)**.** *A probabilisitically checkable proof (*PCP*) for a language $\mathcal{L}$ is a triple of algorithms $(\Pi, \mathcal{Q}_{\mathsf{PCP}}, \mathcal{V}_{\mathsf{PCP}})$ with the following syntax:*

- *Π is a deterministic algorithm that takes as input an instance $x \in \mathcal{L}$ (and possibly some additional information, such as a witness), and outputs a proof string π. We denote by $L = |\pi|$.*

[5] We focus on decisional assumptions for simplicity, and because our reductions are from decisional assumptions.
[6] It will be clear what the θ-decisional complexity assumption is in each context.

- *$\mathcal{Q}_{\mathsf{PCP}}$ is a probabilistic query generation algorithm which takes as input a security parameter 1^κ, and generates a set of queries $q_1, \ldots, q_\ell \in [L]$.*
- *$\mathcal{V}_{\mathsf{PCP}}$ is a deterministic polynomial-time verification algorithm that takes as input an instance x, a set of queries $(q_1, \ldots, q_\ell)$ and a corresponding set of answers $(a_1, \ldots, a_\ell)$, and outputs 0 (reject) or 1 (accept).*

We require the following properties to hold:

1. **(Perfect) Completeness:** *For every $x \in \mathcal{L}$,*

$$\Pr[\mathcal{V}_{\mathsf{PCP}}(x, (q_1, \ldots, q_\ell), (\pi_{q_1}, \ldots, \pi_{q_\ell})) = 1] = 1 \ ,$$

   where $\pi = \Pi(x)$, and where the probability is over $(q_1, \ldots, q_\ell) \leftarrow \mathcal{Q}_{\mathsf{PCP}}(1^\kappa)$.

2. **Soundness:** *For every $x \notin \mathcal{L}$, and for every (possibly malicious) string $\pi^* \in \{0,1\}^*$,*

$$\Pr[\mathcal{V}_{\mathsf{PCP}}(x, (q_1, \ldots, q_\ell), (\pi^*_{q_1}, \ldots, \pi^*_{q_\ell})) = 1] \leq 2^{-\kappa} \ ,$$

   where the probability is over $(q_1, \ldots, q_\ell) \leftarrow \mathcal{Q}_{\mathsf{PCP}}(1^\kappa)$.

We will be interested in PCP's with an additional property, that each query set $Q = (q_1, \ldots, q_\ell) \in \mathcal{Q}_{\mathsf{PCP}}$ can be partitioned into several *tests*, such that the verifier's checks are simply the conjunction of checking each test. This property holds for all PCP's known to the authors.

Definition 2.4. *We say that a* PCP $(\Pi, \mathcal{Q}_{\mathsf{PCP}}, \mathcal{V}_{\mathsf{PCP}})$ is verified via tests *if there is some algorithm $\mathcal{U}_{\mathsf{PCP}}$ such that each query set $Q = (q_1, \ldots, q_\ell) \in \mathcal{Q}_{\mathsf{PCP}}(1^\kappa)$ can be partitioned into θ* tests *$\zeta_1 \cup \cdots \cup \zeta_\theta$, where for every $j \in [\theta]$ there exists a set of indices $I_j \subseteq [\ell]$ such that $\zeta_j = Q|_{I_j}$, and the* PCP *verifier accepts a set of answers $A = (a_1, \ldots, a_\ell)$ if and only if $\mathcal{U}_{\mathsf{PCP}}(x, Q|_{I_j}, A|_{I_j}) = 1$ for every $j \in [\theta]$.*

Remark 2.5. *We also consider a stronger notion of* PCP *soundness known as* non-signaling soundness*, and more specifically* computational non-signaling soundness*. The precise definition is not needed in order to understand our result, since we use it in a black box way. We remark that there is a parameter Ω associated with the computational non-signaling soundness, such that for every $\Omega_1 < \Omega_2$, a Ω_2-computational non-signaling* PCP *is also a Ω_1-computational non-signaling* PCP*. Furthermore, each such* PCP *is associated with a locality parameter ℓ, which for simplicity can be thought of as the query complexity. We refer the reader to* [KRR14, BHK17] *for the precise definitions.*

Computational non-signaling PCP's have been constructed for several classes of languages. One is the language $\mathcal{L}_{\mathcal{U}}(t) = \{\mathcal{L}_{\mathcal{U}}(t(n))\}_{n \in \mathbb{N}}$, where $\mathrm{poly}(n) \leq t(n) \leq \exp(n)$, such that for any (deterministic) Turing machine M and input x, $(M, x) \in \mathcal{L}_{\mathcal{U}}(t)$ if and only if M on input x outputs 1 within $t(|(M, x)|)$ time steps.

Theorem 2.6 ([KRR14, BHK17])**.** *For any $\mathrm{poly}(n) \leq t(n) \leq \exp(n)$, there exists a t-computational non-signaling* PCP *for $\mathcal{L}_{\mathcal{U}}(t)$ with locality $\ell = \kappa \cdot \mathrm{polylog}(t)$, where the* PCP *proof*

has size $L(n) = \mathrm{poly}(t(n))$ *and can be generated in time* $\mathrm{poly}(t(n))$*. Furthermore,* $\mathcal{Q}_{\mathsf{PCP}}(1^\kappa)$ *runs in time* $\mathrm{poly}(\ell)$*, and* $\mathcal{V}_{\mathsf{PCP}}$*, on input* (M, x)*,* $(q_1, \ldots, q_\ell)$*, and* $(a_1, \ldots, a_\ell)$*, runs in time* $|(M, x)| \cdot \mathrm{poly}(\ell)$*.*

Moreover, this PCP *is verified via tests, with a total of* $\mathrm{poly}(t)$ *many possible tests* ζ *(see Definition 2.4).*

Another language with a computational non-signaling PCP is $\mathsf{NL}_{\mathcal{U}}(t, s)$, the class of problems that can be solved nondeterministically in time t and space s. That is, $(M, x) \in \mathsf{NL}_{\mathcal{U}}(t, s)$ if M is a non-deterministic Turing machine that, on input x, runs in space $s(|(M, x)|)$ and outputs 1 within $t(|(M, x)|)$ time steps.

Theorem 2.7 ([BKK+18])**.** *For* $\mathrm{poly}(n) \leq t \leq \exp(n)$ *and* $s = s(n) \geq \log t(n)$*, there is a* 2^s*-computational non-signaling* PCP *for* $\mathsf{NL}_{\mathcal{U}}(t, s)$ *with locality* $\ell = \kappa \cdot \mathrm{poly}(s)$*. The* PCP *proof has size* $L(n) = \mathrm{poly}(t(n))$ *and can be generated in time* $t(n)$*. Furthermore, the query generation algorithm runs in time* $\mathrm{poly}(\ell)$ *and the verifier, on input* $(M, x), (q_1, \ldots, q_\ell), (a_1, \ldots, a_\ell)$*, runs in time* $|(M, x)| \cdot \mathrm{poly}(\ell)$*.*

Moreover, this PCP *is verified via tests. There are a total of* $\mathrm{poly}(t)$ *possible tests* ζ*.*

Finally, [BHK17] constructed a computational non-signaling PCP for all languages in BatchNP, where each language in BatchNP is associated with an NP language L and a batch parameter N, and is denoted by $\mathsf{L}^{\otimes N}$. Each instance in $\mathsf{L}^{\otimes N}$ is a tuple $(x_1, \ldots, x_N)$ such that $x_i \in \mathsf{L}$ $\forall i \in [N]$, $|x_1| = \ldots = |x_N|$, and $N = N(|x_i|)$. In what follows, we denote by $m = |x_i|$.

Theorem 2.8 ([BHK17])**.** *For any* NP *language* L *and* $N = N(m)$*, there is a* $m \cdot N$*-computational non-signaling* PCP *for* $\mathsf{L}^{\otimes N}$ *with locality* $\ell = \kappa \cdot \mathrm{poly}(m, \log N)$*. The* PCP *proof has size* $L = \mathrm{poly}(m, N)$ *and can be generated in time* $\mathrm{poly}(m, N)$*. Furthermore, the query genration algorithm runs in time* $\mathrm{poly}(\ell)$ *and the verifier, on input* $(x_1, \ldots, x_N), (q_1, \ldots, q_\ell), (a_1, \ldots, a_\ell)$*, runs in time* $(N + \mathrm{poly}(\ell)) \cdot \mathrm{poly}(m, \log N)$*.*

2.3 Hash Function Families with Local Opening

In what follows, we assume $L \leq 2^\kappa$.

Definition 2.9 (Hash Family)**.** *A* hash family *is a pair of* PPT *algorithms* $(\mathsf{Gen}, \mathsf{Hash})$*, where*

- $\mathsf{Gen}(1^\kappa, L)$ *takes as input a security parameter* κ *in unary and an input length* L*, and outputs a hash key* $\mathsf{hk} \in \{0,1\}^{\ell_{\mathsf{hk}}}$*.*
- $\mathsf{Hash}(\mathsf{hk}, x)$ *takes as input a hash key* $\mathsf{hk} \in \{0,1\}^{\ell_{\mathsf{hk}}}$ *and an input* $x \in \{0,1\}^L$ *and outputs an element* $\mathsf{rt} \in \{0,1\}^{\ell_{\mathsf{hash}}}$*.*

Here, $\ell_{\mathsf{hk}} = \ell_{\mathsf{hk}}(\kappa) = \mathrm{poly}(\kappa)$ *and* $\ell_{\mathsf{hash}} = \ell_{\mathsf{hash}}(\kappa) = \mathrm{poly}(\kappa)$ *are parameters associated with the hash family.*

Definition 2.10 (Hash Family with Local Opening)**.** *A* hash family with local opening *is a hash family* $(\mathsf{Gen}, \mathsf{Hash})$*, along with two additional* PPT *algorithms* $(\mathsf{Open}, \mathsf{Verify})$ *with the following syntax:*

- $\mathsf{Open}(\mathsf{hk}, x, j)$ *takes as input a hash key* $\mathsf{hk} \in \{0,1\}^{\ell_{\mathsf{hk}}}$*,* $x \in \{0,1\}^L$*, and an index* $j \in [L]$ *and outputs an opening* $\mathsf{o} \in \{0,1\}^{\ell_{\mathsf{o}}}$*, where* $\ell_{\mathsf{o}} = \ell_{\mathsf{o}}(\kappa) = \mathrm{poly}(\kappa)$*.*

- $\mathsf{Verify}(\mathsf{hk}, \mathsf{rt}, j, u, \mathsf{o})$ *takes as input a hash key* $\mathsf{hk} \in \{0,1\}^{\ell_{\mathsf{hk}}}$*, a hash value* $\mathsf{rt} \in \{0,1\}^{\ell_{\mathsf{hash}}}$*, an index* $j \in [L]$*, a value* $u \in \{0,1\}$*, and an opening* $\mathsf{o} \in \{0,1\}^{\ell_{\mathsf{o}}}$*, and outputs* 1 *or* 0 *indicating accept or reject, respectively.*

These algorithms should satisfy the property:

- **Correctness of Opening:** *For every* $x \in \{0,1\}^L$ *and* $j \in [L]$*,*

$$\Pr[\mathsf{Verify}(\mathsf{hk}, \mathsf{Hash}(\mathsf{hk}, x), j, x_j, \mathsf{Open}(\mathsf{hk}, x, j)) = 1] = 1,$$

  where the probability is over $\mathsf{hk} \leftarrow \mathsf{Gen}(1^\kappa, L)$*.*

In our construction, we will require hash functions with local openings that have an additional special property, that for any index $i \in [L]$ one can generate a hash key such that a hash value is statistically binding at position i: namely, all the preimages of a hashed value have the same value at position i. These hash functions are called *somewhere statistically binding* and were first defined in [HW15]. We will also require that the statistically bound value at position i can be recovered via an invert function that is efficient with the aid of a trapdoor.[7]

Definition 2.11 (SSB Hash Family)**.** *A* $S = S(\kappa)$-hiding somewhere statistically binding (SSB) hash family *is a hash family with local opening* $(\mathsf{Gen}, \mathsf{Hash}, \mathsf{Open}, \mathsf{Verify})$*, where*

- $\mathsf{Gen}(1^\kappa, L, i)$ *takes as additional input an index* $i \in [L]$ *and outputs a hash key* $\mathsf{hk} \in \{0,1\}^{\ell_{\mathsf{hk}}}$ *as well as a trapdoor* $\mathsf{td} \in \{0,1\}^{\ell_{\mathsf{td}}}$*,*

along with an additional PPT *algorithm* Invert*:*

- $\mathsf{Invert}(\mathsf{td}, \mathsf{rt})$ *takes as input a trapdoor* $\mathsf{td} \in \{0,1\}^{\ell_{\mathsf{td}}}$ *and a hash value* $\mathsf{rt} \in \{0,1\}^{\ell_{\mathsf{hash}}}$*, and outputs a value* $u \in \{0,1,\perp\}$*.*

These algorithms should satisfy the following properties:

- **S-Index Hiding:** *For any* $\mathrm{poly}(S(\kappa))$*-size adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *there exists a negligible function* μ *such that for any* $L \leq 2^\kappa$*,*

$$\Pr\left[b = b' \;\middle|\; \begin{array}{r} i_1, i_2, \mathsf{state} \leftarrow \mathcal{A}_1(1^\kappa) \\ b \xleftarrow{\$} \{0,1\} \\ (\mathsf{hk}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\kappa, L, i_b) \\ b' \leftarrow \mathcal{A}_2(\mathsf{hk}, \mathsf{state}) \end{array}\right] = \frac{1}{2} + \mu(S(\kappa)).$$

- **Correctness of Inversion:** *For any* $\kappa \in \mathbb{N}$*,* $L \leq 2^\kappa$*, and any* $i \in [L]$ *and* $x \in \{0,1\}^L$*,*

$$\Pr[\mathsf{Invert}(\mathsf{td}, \mathsf{Hash}(\mathsf{hk}, x)) = x_i] = 1,$$

  where the probability is over $(\mathsf{hk}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\kappa, L, i)$*.*

[7] This additional function Invert was not part of the definition of a SSB hash family originally given in [HW15], but their construction satisfies this property.

- **Somewhere Statistically Binding:**[8] *For any* $\kappa \in \mathbb{N}$, $L \leq 2^\kappa$, $i \in [L]$ *and* $\mathsf{rt} \in \{0,1\}^{\ell_{\mathsf{hash}}}$,

$$\Pr[\exists\ u \neq \mathsf{Invert}(\mathsf{td}, \mathsf{rt}), \mathsf{o}\ \textit{s.t.}\ \mathsf{Verify}(\mathsf{hk}, \mathsf{rt}, i, u, \mathsf{o}) = 1] = 0,$$

where the probability is over $(\mathsf{hk}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\kappa, L, i)$.

Hubáček and Wichs constructed SSB hash functions assuming the existence of a leveled homomorphic encryption scheme, and their construction satisfies our stronger variant with trapdoor opening as well.

Theorem 2.12 ([HW15]). *Assuming the sub-exponential hardness of the learning with errors (*LWE*) problem, there exists a* 2^{κ^ϵ}*-hiding* SSB *hash family for some* $\epsilon > 0$. *The* 2^{κ^ϵ}*-hiding is via a* 2^{κ^ϵ}*-straight-line reduction from the* 2^{κ^ϵ}*-hardness of* LWE *(see Definition 2.1).*

2.4 Kilian's Protocol.

Kilian's transformation uses a hash family with local opening and a PCP scheme to construct a 4-round succinct argument.

For our description of Kilian's protocol, fix any hash family with local opening $\mathcal{H} = (\mathsf{Gen}, \mathsf{Hash}, \mathsf{Open}, \mathsf{Verify})$ and a PCP scheme $(\Pi, \mathcal{Q}_{\mathsf{PCP}}, \mathcal{V}_{\mathsf{PCP}})$ for a language $\mathcal{L}$. Denote the length of a PCP proof by $L = L(n)$. Kilian's protocol is given in Figure 1.

Kilian's Protocol

On input x and security parameter 1^κ, the 4-message protocol $(\mathcal{P}_{\mathsf{Kilian}}, \mathcal{V}_{\mathsf{Kilian}})$ proceeds as follows.

- **First verifier's message:** $\mathcal{V}_{\mathsf{Kilian}}$ samples $\mathsf{hk} \leftarrow \mathsf{Gen}(1^\kappa, L)$, and sends hk to the prover.
- **First prover's meessage:** $\mathcal{P}_{\mathsf{Kilian}}$ computes the PCP proof $\pi = \Pi(x)$, and its hash value $\mathsf{rt} = \mathsf{Hash}(\mathsf{hk}, \pi)$. It sends rt to the verifier.
- **Second verifier's message:** $\mathcal{V}_{\mathsf{Kilian}}$ computes a set of queries $(q_1, \ldots, q_\ell) \leftarrow \mathcal{Q}_{\mathsf{PCP}}(1^\kappa)$, and sends $(q_1, \ldots, q_\ell)$ to the prover.
- **Second prover's message:** $\mathcal{P}_{\mathsf{Kilian}}$ computes for every $i \in [\ell]$ the opening $\mathsf{o}_i = \mathsf{Open}(\mathsf{hk}, \pi, q_i)$, and sends $\{\pi_{q_i}, \mathsf{o}_i\}_{i \in [\ell]}$ to the verifier.
- **Verdict:** $\mathcal{V}_{\mathsf{Kilian}}$ accepts if and only if $\mathcal{V}_{\mathsf{PCP}}(x, (q_1, \ldots, q_\ell), (\pi_{q_1}, \ldots, \pi_{q_\ell})) = 1$ and for every $i \in [\ell]$, $\mathsf{Verify}(\mathsf{hk}, \mathsf{rt}, q_i, \pi_{q_i}, \mathsf{o}_i) = 1$.

Figure 1: Kilian's Protocol $(\mathcal{P}_{\mathsf{Kilian}}, \mathcal{V}_{\mathsf{Kilian}})$ for a Language $\mathcal{L}$

[8] We remark that our definition of somewhere statistically binding is different and slightly stronger than the original notion given in [HW15], which states that for any $\kappa \in \mathbb{N}$, $L \in \mathbb{N}$, $i \in [L]$ and $\mathsf{rt} \in \{0,1\}^{\ell_{\mathsf{hash}}}$, $\Pr[\exists\ u \neq u', \mathsf{o}, \mathsf{o}'$ s.t. $\mathsf{Verify}(\mathsf{hk}, \mathsf{rt}, i, u, \mathsf{o}) = \mathsf{Verify}(\mathsf{hk}, \mathsf{rt}, i, u', \mathsf{o}') = 1] = 0$, where the probability is over $\mathsf{hk} \leftarrow \mathsf{Gen}(1^\kappa, L, i)$. The difference is that in our definition, there are certain "invalid" hash values (for which Invert outputs $\perp$) which should have no valid openings, but in theirs, they simply require that there are at most one valid opening for every hash value.

2.5 The BMW Heuristic.

The BMW heuristic converts a PCP scheme into a 2-message, succinct, privately verifiable argument. It does this by allowing one to query a PCP proof using a private information retrieval (PIR) scheme, which we define below.

Definition 2.13 ([CGKS95, KO97]). *A* 1-server private information retrieval (PIR) scheme *is a tuple of* PPT *algorithms* $(\mathsf{Query}, \mathsf{Answer}, \mathsf{Reconstruct})$ *with the following syntax:*

- $\mathsf{Query}(1^\kappa, L, q)$ *takes as input a security parameter* κ *in unary, an input size* L*, and an index* $q \in [L]$*, and outputs a query* $\hat{q}$ *along with a trapdoor* td*.*
- $\mathsf{Answer}(\hat{q}, x)$ *takes as input a query* $\hat{q}$ *and a database* $x \in \{0,1\}^L$*, and outputs an answer* $\hat{a}$*.*
- $\mathsf{Reconstruct}(\mathsf{td}, \hat{a})$ *takes as input a trapdoor* td *and an answer* $\hat{a}$*, and outputs a plaintext* a*.*

These algorithms should satisfy the following properties:

- **Correctness:** *For every* $\kappa, L \in \mathbb{N}$ *and* $q \in [L]$*,*

$$\Pr[\mathsf{Reconstruct}(\mathsf{td}, \mathsf{Answer}(\hat{q}, x)) = x_q] = 1,$$

  where the probability is over $(\hat{q}, \mathsf{td}) \leftarrow \mathsf{Query}(1^\kappa, q, L)$*.*

- S**-Privacy:** *For any* $\mathrm{poly}(S(\kappa))$*-size adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *there exists a negligible function* μ *such that for every* $\kappa, L \in \mathbb{N}$*,*

$$\Pr\left[b = b' \;\middle|\; \begin{array}{r} q_0, q_1, \mathsf{state} \leftarrow \mathcal{A}_1(1^\kappa, L) \\ b \xleftarrow{\$} \{0,1\} \\ (\hat{q}, \mathsf{td}) \leftarrow \mathsf{Query}(1^\kappa, L, q_b) \\ b' \leftarrow \mathcal{A}_2(\hat{q}, \mathsf{state}) \end{array}\right] = \frac{1}{2} + \mu(S(\kappa)).$$

Kushilevitz and Ostrovsky [KO97] constructed the first sublinear-communication single-server PIR scheme and was followed up by several other works [GR05, Lip05, BV11, DGI+19].

Theorem 2.14 ([BV11, DGI+19]). *For any function* $S : \mathbb{N} \to \mathbb{N}$*, there exists a* S*-private 1-server* PIR *scheme with* $\mathrm{polylog}(L)$ *query complexity for length-*L *databases, under the* S*-hardness of the* LWE*, Quadratic Residuosity, or* DDH *assumptions. Moreover, these schemes are* S*-straight-line secure (see Defintion 2.1).*

Fix any 1-server PIR scheme $(\mathsf{Query}, \mathsf{Answer}, \mathsf{Reconstruct})$ and any PCP scheme $(\Pi, \mathcal{Q}_{\mathsf{PCP}}, \mathcal{V}_{\mathsf{PCP}})$ for a language $\mathcal{L}$. The BMW heuristic is a 2-message succinct argument for $\mathcal{L}$, defined in Figure 2.

Theorem 2.15 ([KRR14, BHK17]). *The protocol* $(\mathcal{P}_{\mathsf{BMW}}, \mathcal{V}_{\mathsf{BMW}})$ *has the following guarantees:*

The BMW Protocol

On input 1^κ and x, the 2-message protocol $(\mathcal{P}_{\mathsf{BMW}}, \mathcal{V}_{\mathsf{BMW}})$ proceeds as follows:

- **Verifier:** $\mathcal{V}_{\mathsf{BMW}}$ computes $(q_1, \ldots, q_\ell) \leftarrow \mathcal{Q}_{\mathsf{PCP}}(1^\kappa)$. For each $i \in [\ell]$, it generates $(\hat{q}_i, \mathsf{td}_i) \leftarrow \mathsf{Query}(1^\kappa, L, q_i)$, where L is the length of the PCP. It sends $\{\hat{q}_i\}_{i\in\ell}$ to the prover.
- **Prover:** $\mathcal{P}_{\mathsf{BMW}}$ computes the PCP string $\pi = \Pi(x)$, and for each $i \in [\ell]$, it computes $\hat{a}_i = \mathsf{Answer}(\hat{q}_i, \pi)$. It sends $\{\hat{a}_i\}_{i\in[\ell]}$ to the verifier.
- **Verdict:** $\mathcal{V}_{\mathsf{BMW}}$ computes $a_i = \mathsf{Reconstruct}(\mathsf{td}_i, \hat{a}_i)$ for each $i \in [\ell]$, and accepts if and only if $V_{\mathsf{PCP}}(x, (q_1, \ldots, q_\ell), (a_1, \ldots, a_\ell)) = 1$.

Figure 2: The BMW Protocol $(P_{\mathsf{BMW}}, V_{\mathsf{BMW}})$ for $\mathcal{L}$

- **Completeness:** *For every $x \in \mathcal{L}$,*

$$\Pr[(\mathcal{P}_{\mathsf{BMW}}, \mathcal{V}_{\mathsf{BMW}})(x) = 1] = 1,$$

where the probability is over the random coin tosses of $\mathcal{V}_{\mathsf{BMW}}$.

- **Straight-Line Soundness:** *Suppose the underlying* PCP *$(\Pi, \mathcal{Q}_{\mathsf{PCP}}, \mathcal{V}_{\mathsf{PCP}})$ has $\Omega(n)$-computational non-signaling soundness (see Remark 2.5). Let $\Omega' = \Omega'(\kappa)$ be such that $\Omega'(\kappa) = \Omega(n)$, and assume that $2^{-\kappa} = \mathrm{negl}(\Omega')$. Then, assuming the* PIR *scheme has Ω'-privacy, for every $\mathrm{poly}(\Omega')$-size cheating prover $\mathcal{P}^*$, there exists a negligible function μ such that for every $x \notin \mathcal{L}$,*

$$\Pr[(\mathcal{P}^*, \mathcal{V}_{\mathsf{BMW}})(x) = 1] \leq \mu(\Omega').$$

Moreover, the soundness is Ω'-straight-line (Definition 2.1).[9]

3 Somewhere Statistically Sound Interactive Arguments

Definition 3.1. *An interactive argument $(\mathcal{P}, \mathcal{V})$ for a language $\mathcal{L} = \{\mathcal{L}_n\}_{n\in\mathbb{N}}$ is* θ-statistically sound *if for every (unbounded) cheating prover $\mathcal{P}^*$ there exists a negligible function μ such that for every $x \notin \mathcal{L}$,*

$$\Pr[(\mathcal{P}^*, \mathcal{V})(1^\kappa, x) = 1] \leq \mu(\theta).$$[10]

Definition 3.2. *An interactive argument $(\mathcal{P}, \mathcal{V})(1^\kappa)$ for a language $\mathcal{L} = \{\mathcal{L}_n\}_{n\in\mathbb{N}}$ is $\theta = \theta(\kappa)$-*somewhere statistically sound (SSS) *with respect to a θ-decisional complexity assumption* A *if for every first verifier message β_1, there exists a second verifier message $T(\beta_1)$ such that:*

[9] Note that the Ω'-privacy of the PIR scheme is a Ω'-decisional assumption. Moreover, if we use one of the PIR schemes from Theorem 2.14 then it further reduces (via a straight-line reduction) to a standard Ω'-decisional assumption, such as Ω'-hardness of LWE, QR, or DDH.

[10] The definition may seem slightly convoluted, and that a more natural definition would require that $\Pr[(\mathcal{P}^*, \mathcal{V})(1^\kappa, x) = 1] \leq \theta$. We chose the definition since it composes nicely with the definition of somewhere statistical soundness (Definition 3.2).

- **θ-Somewhere Statistically Sound:** *For every $\mathrm{poly}(\theta)$-size (cheating) prover $\mathcal{P}^*$, conditioned on the first three messages being $(\beta_1, \mathcal{P}^*(\beta_1), T(\beta_1))$, the remaining protocol is θ-statistically sound with overwhelming probability $1 - \mathrm{negl}(\theta)$ over β_1, assuming* A.

  Moreover, this condition holds in a θ-straight-line manner; i.e., there is a black box reduction $\mathcal{R}$ such that $\mathcal{R}$, given oracle access to a cheating prover $\mathcal{P}^*$ for which the protocol beginning with $(\beta_1, \mathcal{P}^*(\beta_1), T(\beta_1))$ is not θ-statistically sound with overwhelming probability $1 - \mathrm{negl}(\theta)$, simulates the protocol with the prover by sending a message for every round once (without rewinding), where the messages for the first two verifier rounds are β_1 and $T(\beta_1)$, and uses the resulting transcript to break the underlying assumption A.

- **θ-Computational Indistinguishability:** *For any $\mathrm{poly}(\theta)$-size distinguisher $\mathcal{D}$,*

$$\left| \Pr_{\beta_1}[\mathcal{D}(\beta_1, T(\beta_1)) = 1] - \Pr_{\beta_1, \beta_2}[\mathcal{D}(\beta_1, \beta_2) = 1] \right| \leq \mathrm{negl}(\theta).$$

  Furthermore, this indistinguishability is θ-straight line, with respect to assumption A.

We remark that this is a strong definition: our cheating prover proceeds in two stages, a stage-1 $\mathcal{P}_1^*$ which is computationally bounded and produces the second message; and a stage-2 $\mathcal{P}_2^*$ who produces the rest of the transcript, and has no computational limitations. How could one possibly use a cheating prover $(\mathcal{P}_1^*, \mathcal{P}_2^*)$ to break a computational assumption when $\mathcal{P}_2^*$ is unbounded? While this seems mysterious at first sight, we remark that similar situations arise in other places, e.g., in the proof of the [KRR14] protocol. Indeed, we will use similar ideas in our reduction.

Theorem 3.3. *Any θ-*SSS *interactive argument $(\mathcal{P}, \mathcal{V})$ w.r.t. a θ-decisional complexity assumption* A *is θ-straight-line sound.*

Proof. To prove straight-line soundness, we will define a straight-line reduction from the θ-somewhere statistically sound and θ-computational indistinguishability assumptions to the θ-soundness of $(\mathcal{P}, \mathcal{V})$. Then, combining with the fact that there is a straight-line reduction from some θ-decisional complexity assumption A to the θ-somewhere statistically sound and θ-computational indistinguishability properties, we obtain that there is a θ-straight-line reduction from A to the θ-soundness of $(\mathcal{P}, \mathcal{V})$.

Suppose that there is a $\mathrm{poly}(\theta)$-size cheating prover $\mathcal{P}^*$ and $x \notin \mathcal{L}$ such that $\Pr[(\mathcal{P}^*, \mathcal{V})(x) = 1] = \delta(\theta)$, where δ is a non-negligible function. Now, given (β_1, β_2), in which either $\beta_2 = T(\beta_1)$ or β_2 is random, reduction $\mathcal{R}$ simulates an interaction of $\mathcal{V}$ with $\mathcal{P}^*$ using the first two verifier messages β_1 and β_2. If the resulting transcript is accepting, $\mathcal{R}$ outputs 1. Otherwise, it outputs 0.

Note that

$$\Pr_{\beta_1, \beta_2}[(\mathcal{P}^*, \mathcal{V})(x) = 1] = \delta(\theta),$$

so the distinguishing advantage of the reduction is

$$\delta(\theta) - \Pr_{\beta_1}[(\mathcal{P}^*, \mathcal{V})(x) = 1 \mid \beta_2 = T(\beta_1)],$$

which under the θ-somewhere statistically sound assumption is $\delta(\theta) - \mathrm{negl}(\theta)$, which is non-negligible in θ. This means that the θ-somewhere statistically sound and θ-computationally indistinguishability properties cannot simultaneously hold. □

Finally, we prove that the importance of a θ-straight-line sound argument is that if the underlying θ-decisional complexity assumption is θ-post-quantum secure, then the argument is sound against $\mathrm{poly}(\theta)$-size quantum provers, with overwhelming probability in θ.

Theorem 3.4. *Any argument $(\mathcal{P}, \mathcal{V})$ that is θ-straight-line sound w.r.t. a θ-decisional complexity assumption* A*, is also post-quantum sound assuming* A *holds w.r.t. quantum adversaries.*

Proof. Fix any $\mathrm{poly}(\theta)$-size cheating quantum prover $\mathcal{P}^*$ that for infinitely many $\kappa \in \mathbb{N}$, convinces $\mathcal{V}$ of a rejecting instance with probability $1/\mathrm{poly}(\theta(\kappa))$. By the θ-straight-line soundness, there exists a PPT black-box reduction $\mathcal{R}$ that given oracle access to any classical cheating prover $\mathcal{P}^{**}$ that breaks soundness with probability $1/\mathrm{poly}(\theta)$, interacts with $\mathcal{P}^{**}$ *once* (without rewinding) by sending $\mathcal{P}^{**}$ a single message for each round, and using the transcript obtained, breaks assumption A.

We next argue that $\mathcal{R}$ successfully breaks A even given oracle access to the quantum adversary $\mathcal{P}^*$. This follows from the following observations. First, observe that $\mathcal{R}$ interacts with $\mathcal{P}^*$ using completely classical messages. Secondly, $\mathcal{P}^*$ can be simulated exactly by an unbounded classical adversary $\mathcal{P}^{**}$, which therefore also generates an accepting transcript with probability $1/\mathrm{poly}(\theta)$. Finally, since the reduction is straight-line, it cannot distinguish between having oracle access to $\mathcal{P}^*$ and having oracle access to $\mathcal{P}^{**}$. Put together, since the reduction with oracle access to $\mathcal{P}^{**}$ breaks A, it also breaks A given (non-rewinding) oracle access to $\mathcal{P}^*$. □

4 Kilian's Protocol is Somewhere Statistically Sound

In this section, we show that Kilian's protocol, when instantiated with a computational non-signaling PCP and a specific hash family we call ℓ-SSB, is a SSS interactive argument.

4.1 The BMW Heuristic with SSB Hash Families

The key idea in our instantiation of Kilian's protocol is that the first two messages in Kilian's protocol can be viewed as a run of the BMW protocol. Specifically, the hash function we use in Kilian's protocol is several SSB hash functions, each of which acts as a PIR scheme, as follows:

- $\mathsf{Query}(1^\kappa, L, q)$ calls $(\mathsf{hk}_{\mathsf{SSB}}, \mathsf{td}_{\mathsf{SSB}}) \leftarrow \mathsf{Gen}_{\mathsf{SSB}}(1^\kappa, L, q)$ and outputs $(\hat{q}, \mathsf{td})$, where $\hat{q} = \mathsf{hk}_{\mathsf{SSB}}$ and $\mathsf{td} = \mathsf{td}_{\mathsf{SSB}}$.
- $\mathsf{Answer}(\hat{q}, \pi)$ takes as input $\hat{q} = \mathsf{hk}_{\mathsf{SSB}}$ and $\pi \in \{0,1\}^L$ and produces $\hat{a} = \mathsf{rt} = \mathsf{Hash}_{\mathsf{SSB}}(\mathsf{hk}_{\mathsf{SSB}}, \pi)$.
- $\mathsf{Reconstruct}(\mathsf{td}, \hat{a})$ takes as input $\mathsf{td} = \mathsf{td}_{\mathsf{SSB}}$ and $\hat{a} = \mathsf{rt}$ and outputs $\mathsf{Invert}_{\mathsf{SSB}}(\mathsf{td}_{\mathsf{SSB}}, \mathsf{rt})$.

Claim 4.1. $(\mathsf{Gen}_{\mathsf{SSB}}, \mathsf{Hash}_{\mathsf{SSB}}, \mathsf{Invert}_{\mathsf{SSB}})$ *is an S-private* PIR *scheme if* $(\mathsf{Gen}_{\mathsf{SSB}}, \mathsf{Hash}_{\mathsf{SSB}}, \mathsf{Open}_{\mathsf{SSB}}, \mathsf{Verify}_{\mathsf{SSB}}, \mathsf{Invert}_{\mathsf{SSB}})$ *is an S-hiding* SSB *hash family.*

Proof. The correctness condition of the PIR scheme follows from the correctness of $\mathsf{Invert}_{\mathsf{SSB}}$, and the PIR scheme has S-privacy if the SSB hash family is S-hiding. □

Note that the first (verifier) message of Kilian's protocol is the choice of a single hash function, while the first message of the BMW protocol consists of many PIR queries (or in our case, SSB hash keys) corresponding to the many locations that the PCP verifier accesses the PCP string. We will reconcile the two by defining a new hash family, in which each hash function consists of ℓ parallel SSB hash functions, to use in Kilian's protocol. We call this hash family a *ℓ-somewhere statistically binding hash family (ℓ-SSB)*, and remark that a straightforward repetition of an SSB hash family gives us ℓ-SSB. For completeness, we write down the construction in Figure 3 and mention some of its properties.

An ℓ-SSB Hash Family

Let $\mathcal{H}_{\mathsf{SSB}} = (\mathsf{Gen}_{\mathsf{SSB}}, \mathsf{Hash}_{\mathsf{SSB}}, \mathsf{Open}_{\mathsf{SSB}}, \mathsf{Verify}_{\mathsf{SSB}}, \mathsf{Invert}_{\mathsf{SSB}})$ be an S-hiding SSB hash family. The ℓ-SSB hash family

$$\mathcal{H}_{\ell\text{-}\mathsf{SSB}} = (\mathsf{Gen}_{\ell\text{-}\mathsf{SSB}}, \mathsf{Hash}_{\ell\text{-}\mathsf{SSB}}, \mathsf{Open}_{\ell\text{-}\mathsf{SSB}}, \mathsf{Verify}_{\ell\text{-}\mathsf{SSB}}, \mathsf{Invert}_{\ell\text{-}\mathsf{SSB}}),$$

is defined as follows:

- $\mathsf{Gen}_{\ell\text{-}\mathsf{SSB}}(1^\kappa, L, (i_1, \ldots, i_\ell))$ samples for every $j \in [\ell]$ a pair $(\mathsf{hk}_{\mathsf{SSB},j}, \mathsf{td}_{\mathsf{SSB},j}) \leftarrow \mathsf{Gen}_{\mathsf{SSB}}(1^\kappa, L, i_j)$. It outputs $\mathsf{hk}_{\ell\text{-}\mathsf{SSB}} = (\mathsf{hk}_{\mathsf{SSB},1}, \ldots, \mathsf{hk}_{\mathsf{SSB},\ell})$ and $\mathsf{td}_{\ell\text{-}\mathsf{SSB}} = (\mathsf{td}_{\mathsf{SSB},1}, \ldots, \mathsf{td}_{\mathsf{SSB},\ell})$.
- $\mathsf{Hash}_{\ell\text{-}\mathsf{SSB}}(\mathsf{hk}_{\ell\text{-}\mathsf{SSB}}, x)$ takes as input a hash key $\mathsf{hk}_{\ell\text{-}\mathsf{SSB}} = (\mathsf{hk}_{\mathsf{SSB},1}, \ldots, \mathsf{hk}_{\mathsf{SSB},\ell})$ and an input $x \in \{0,1\}^L$ and outputs $\mathsf{rt} = (\mathsf{rt}_1, \ldots, \mathsf{rt}_\ell)$, where $\mathsf{rt}_j = \mathsf{Hash}_{\mathsf{SSB}}(\mathsf{hk}_{\mathsf{SSB},j}, x)$ for every $j \in [\ell]$.
- $\mathsf{Open}_{\ell\text{-}\mathsf{SSB}}(\mathsf{hk}_{\ell\text{-}\mathsf{SSB}}, x, k)$ takes as input a hash key $\mathsf{hk}_{\ell\text{-}\mathsf{SSB}} = (\mathsf{hk}_{\mathsf{SSB},1}, \ldots, \mathsf{hk}_{\mathsf{SSB},\ell})$, $x \in \{0,1\}^L$, and an index $k \in [L]$, and outputs the opening $\mathsf{o} = (\mathsf{o}_1, \ldots, \mathsf{o}_\ell)$, where $\mathsf{o}_j \leftarrow \mathsf{Open}_{\mathsf{SSB}}(\mathsf{hk}_{\mathsf{SSB},j}, x, k)$.
- $\mathsf{Verify}_{\ell\text{-}\mathsf{SSB}}(\mathsf{hk}_{\ell\text{-}\mathsf{SSB}}, \mathsf{rt}, k, u, \mathsf{o})$ takes as input a hash key $\mathsf{hk}_{\ell\text{-}\mathsf{SSB}} = (\mathsf{hk}_{\mathsf{SSB},1}, \ldots, \mathsf{hk}_{\mathsf{SSB},\ell})$, $\mathsf{rt} = (\mathsf{rt}_1, \ldots, \mathsf{rt}_\ell)$, $k \in [L]$, $u \in \{0,1\}$, and an opening $\mathsf{o} = (\mathsf{o}_1, \ldots, \mathsf{o}_\ell)$, and outputs 1 if and only if $\mathsf{Verify}_{\mathsf{SSB}}(\mathsf{hk}_{\mathsf{SSB},j}, \mathsf{rt}_j, k, u, \mathsf{o}_j) = 1 \ \forall j \in [\ell]$.
- $\mathsf{Invert}_{\ell\text{-}\mathsf{SSB}}(\mathsf{td}_{\ell\text{-}\mathsf{SSB}}, \mathsf{rt})$ takes as input $\mathsf{td}_{\ell\text{-}\mathsf{SSB}} = (\mathsf{td}_{\mathsf{SSB},1}, \ldots, \mathsf{td}_{\mathsf{SSB},\ell})$ and $\mathsf{rt} = (\mathsf{rt}_1, \ldots, \mathsf{rt}_\ell)$ and outputs the ℓ values $(\mathsf{Invert}_{\mathsf{SSB}}(\mathsf{td}_{\mathsf{SSB},1}, \mathsf{rt}_1), \ldots, \mathsf{Invert}_{\mathsf{SSB}}(\mathsf{td}_{\mathsf{SSB},\ell}, \mathsf{rt}_\ell))$.

Figure 3: The ℓ-SSB Hash Family $(\mathsf{Gen}_{\ell\text{-}\mathsf{SSB}}, \mathsf{Hash}_{\ell\text{-}\mathsf{SSB}}, \mathsf{Open}_{\ell\text{-}\mathsf{SSB}}, \mathsf{Verify}_{\ell\text{-}\mathsf{SSB}}, \mathsf{Invert}_{\ell\text{-}\mathsf{SSB}})$

Lemma 4.2. *The ℓ-SSB hash family $\mathcal{H}_{\ell\text{-}\mathsf{SSB}}$ defined in Figure 3 satisfies the following properties:*

- **S-Index Hiding:** *Assuming $\ell \leq \mathrm{poly}(S(\kappa))$, for any $\mathrm{poly}(S(\kappa))$-size adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ there exists a negligible μ such that for any $\kappa \in \mathbb{N}$,*

$$\Pr\left[b = b' \;\middle|\; \begin{array}{r} i_1^0, \ldots, i_\ell^0, i_1^1, \ldots, i_\ell^1, \mathsf{state} \leftarrow \mathcal{A}_1(1^\kappa) \\ b \xleftarrow{\$} \{0,1\} \\ (\mathsf{hk}_{\ell\text{-}\mathsf{SSB}}, \mathsf{td}_{\ell\text{-}\mathsf{SSB}}) \leftarrow \mathsf{Gen}_{\ell\text{-}\mathsf{SSB}}(1^\kappa, L, \{i_j^b\}_{j\in[\ell]}) \\ b' \leftarrow \mathcal{A}_2(\mathsf{hk}_{\ell\text{-}\mathsf{SSB}}, \mathsf{state}) \end{array}\right] = \frac{1}{2} + \mu(S(\kappa)).$$

Moreover, this holds via a S-straight-line reduction from the S-hiding of the underlying SSB *hash family.*

- **Correctness of Inversion:** *For any $\kappa \in \mathbb{N}$, $L \leq 2^\kappa$, $i_1, \ldots, i_\ell \in [L]$ and $x \in \{0,1\}^L$,*

$$\Pr[\mathsf{Invert}_{\ell\text{-}\mathsf{SSB}}(\mathsf{td}_{\ell\text{-}\mathsf{SSB}}, \mathsf{Hash}(\mathsf{hk}_{\ell\text{-}\mathsf{SSB}}, x)) = (x_{i_1}, \ldots, x_{i_\ell})] = 1.$$

where the probability is over $(\mathsf{hk}_{\ell\text{-}\mathsf{SSB}}, \mathsf{td}_{\ell\text{-}\mathsf{SSB}}) \leftarrow \mathsf{Gen}_{\ell\text{-}\mathsf{SSB}}(1^\kappa, L, \{i_j^b\}_{j\in[\ell]})$.

- **ℓ-Somewhere Statistically Binding:** *For any $\kappa \in \mathbb{N}$, $L \leq 2^\kappa$, $i_1, \ldots, i_\ell \in [L]$, and $\mathsf{rt} \in \{0,1\}^{\ell \cdot \ell_{\mathsf{hash}}}$,*

$$\Pr\left[\begin{array}{l}\exists\ (u_1, \ldots, u_\ell) \neq \mathsf{Invert}_{\ell\text{-}\mathsf{SSB}}(\mathsf{td}_{\ell\text{-}\mathsf{SSB}}, \mathsf{rt}), (\mathsf{o}_1, \ldots, \mathsf{o}_\ell) \\ \quad s.t.\ \mathsf{Verify}_{\ell\text{-}\mathsf{SSB}}(\mathsf{hk}_{\ell\text{-}\mathsf{SSB}}, \mathsf{rt}, i_j, u_j, \mathsf{o}_j) = 1\ \forall j \in [\ell]\end{array}\right] = 0.$$

where the probability is over $(\mathsf{hk}_{\ell\text{-}\mathsf{SSB}}, \mathsf{td}_{\ell\text{-}\mathsf{SSB}}) \leftarrow \mathsf{Gen}_{\ell\text{-}\mathsf{SSB}}(1^\kappa, L, \{i_j\}_{j\in[\ell]})$.

Proof. The correctness of inversion and ℓ-somewhere statistically binding properties follow straightforwardly from the corresponding properties of the underlying SSB hash family (Definition 2.11), so we focus on the S-index hiding property. In particular, we present a straight-line reduction from the S-hiding of the underlying SSB hash family to the S-hiding of the ℓ-SSB hash family.

Suppose that there were a size-$\mathrm{poly}(S)$ algorithm $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ such that for $(i_1^0, \ldots, i_\ell^0, i_1^1, \ldots, i_\ell^1), \mathsf{state} \leftarrow \mathcal{A}_1(1^\kappa)$, $\mathcal{A}_2(\cdot, \mathsf{state})$ can distinguish between $\mathsf{hk}_{\ell\text{-}\mathsf{SSB}}$ generated on index locations $\{i_j^0\}_{j\in[\ell]}$ and $\{i_j^1\}_{j\in[\ell]}$ with probability $\delta(S)$, where δ is a non-negligible function. Fix $(i_1^0, \ldots, i_\ell^0, i_1^1, \ldots, i_\ell^1)$ to be the output of $\mathcal{A}_1$ for which $\mathcal{A}_2$ has the greatest distinguishing advantage, which is at least $\delta(S)$. By a hybrid argument, there is some index j^* for which $\mathcal{A}_2(\cdot, \mathsf{state})$ can distinguish between $\mathsf{hk}_{\ell\text{-}\mathsf{SSB}}$ generated on indices $(i_1^0, \ldots, i_{j^*-1}^0, i_{j^*}^1, i_{j^*+1}^1, \ldots, i_\ell)$ and $(i_1^0, \ldots, i_{j^*-1}^0, i_{j^*}^0, i_{j^*+1}^1, \ldots, i^\ell)$ with probability $\geq \delta(2^{\kappa^\epsilon})/\ell$. Then, to break the S-hiding of the SSB hash family, an adversary can distinguish between $\mathsf{hk}^*_{\mathsf{SSB}}$ generated by $\mathsf{Gen}_{\mathsf{SSB}}(1^\kappa, L, i_{j^*}^0)$ and $\mathsf{Gen}_{\mathsf{SSB}}(1^\kappa, L, i_{j^*}^1)$ by generating $(\mathsf{hk}_{\mathsf{SSB},j}, \mathsf{td}_{\mathsf{SSB},j}) \leftarrow \mathsf{Gen}_{\mathsf{SSB}}(1^\kappa, L, i_j^{b(j)})$ for $j \in [j^* - 1] \cup [j^* + 1, \ell]$, where $b(j) = 1$ if $j > j^*$ and $b(j) = 0$ if $j < j^*$. Then, she runs $\mathcal{A}_2(\cdot, \mathsf{state})$ on the ℓ-SSB hash key $(\mathsf{hk}_{\mathsf{SSB},1}, \ldots, \mathsf{hk}_{\mathsf{SSB},j^*-1}, \mathsf{hk}^*_{\mathsf{SSB}}, \mathsf{hk}_{\mathsf{SSB},j^*+1}, \ldots, \mathsf{hk}_{\mathsf{SSB},\ell})$ and outputs $\mathcal{A}_2(\cdot, \mathsf{state})$'s output. This has a distinguishing advantage of $\geq \delta(2^{\kappa^\epsilon})/\ell$, which is non-negligible in 2^{κ^ϵ}.

Finally, observe that this reduction is straight-line. □

Note that if we instantiate the Kilian protocol with an ℓ-SSB hash family and a PCP with computational non-signaling soundness, then by Theorem 2.15 and Claim 4.1, the statistically committed answers will be rejecting with high probability. This is summarized in the following corollary.

Corollary 4.3 (Corollary of Theorem 2.15)**.** *Let $(\Pi, \mathcal{Q}_{\mathsf{nsPCP}}, \mathcal{V}_{\mathsf{nsPCP}})$ be a* PCP *for $\mathcal{L}$ with $\Omega(n)$-computational non-signaling soundness, and let $\mathcal{H}_{\ell\text{-}\mathsf{SSB}}$ be the ℓ-*SSB *hash family given in Figure 3, where ℓ is the locality, i.e. the size of a* PCP *query generated by $\mathcal{Q}_{\mathsf{nsPCP}}(1^\kappa)$. Assume*

that the underlying SSB *hash family is Ω'-hiding, where $\Omega' = \Omega'(\kappa)$ is such that $\Omega'(\kappa) = \Omega(n)$ and $2^{-\kappa} = \mathrm{negl}(\Omega')$. Then, for any $\mathrm{poly}(\Omega'(\kappa))$-size adversary $\mathcal{A}$ and for any $x \notin \mathcal{L}$,*

$$\Pr\left[\mathcal{V}_{\mathsf{nsPCP}}(x, Q, \mathsf{Invert}_{\ell\text{-}\mathsf{SSB}}(\mathsf{td}_{\ell\text{-}\mathsf{SSB}}, \mathsf{rt})) = 1 \;\middle|\; \begin{array}{r} Q \leftarrow \mathcal{Q}_{\mathsf{nsPCP}}(1^\kappa, L) \\ (\mathsf{hk}_{\ell\text{-}\mathsf{SSB}}, \mathsf{td}_{\ell\text{-}\mathsf{SSB}}) \leftarrow \mathsf{Gen}_{\ell\text{-}\mathsf{SSB}}(1^\kappa, L, Q) \\ \mathsf{rt} \leftarrow \mathcal{A}(x, \mathsf{hk}_{\ell\text{-}\mathsf{SSB}}) \end{array}\right] = \mathrm{negl}(\Omega'). \tag{1}$$

Moreover, this is proven via a Ω'-straight-line reduction (Definition 2.1).[11]

4.2 Kilian with Non-Signaling PCP and ℓ-SSB Hashing

We instantiate Kilian's protocol with two ingredients: a $\Omega(n)$-computational non-signaling PCP $(\Pi, \mathcal{Q}_{\mathsf{nsPCP}}, \mathcal{V}_{\mathsf{nsPCP}})$ for a language $\mathcal{L}$, and the ℓ-SSB hash family $(\mathsf{Gen}_{\ell\text{-}\mathsf{SSB}}, \mathsf{Hash}_{\ell\text{-}\mathsf{SSB}}, \mathsf{Open}_{\ell\text{-}\mathsf{SSB}}, \mathsf{Verify}_{\ell\text{-}\mathsf{SSB}}, \mathsf{Invert}_{\ell\text{-}\mathsf{SSB}})$ given in Figure 3. The resulting protocol is described in Figure 4.

Kilian's protocol instantiated with a Ω-computational non-signaling PCP and an ℓ-SSB hash family

Let $\epsilon \in (0,1)$ be a small constant, and let $\kappa = \kappa(n) = (\log \Omega(n))^{1/\epsilon}$. On input x, the 4-message protocol $(\mathcal{P}_{\mathsf{nsKilian}}, \mathcal{V}_{\mathsf{nsKilian}})$ proceeds as follows.

- **First verifier's message:** $\mathcal{V}_{\mathsf{nsKilian}}$ samples $Q \leftarrow \mathcal{Q}_{\mathsf{nsPCP}}(1^\kappa)$ and $(\mathsf{hk}_{\ell\text{-}\mathsf{SSB}}, \mathsf{td}_{\ell\text{-}\mathsf{SSB}}) \leftarrow \mathsf{Gen}_{\ell\text{-}\mathsf{SSB}}(1^\kappa, L, Q)$, and sends $\mathsf{hk}_{\ell\text{-}\mathsf{SSB}}$ to the prover.
- **First prover's meessage:** $\mathcal{P}_{\mathsf{nsKilian}}$ computes the PCP proof $\pi = \Pi(x)$ and its hash value $\mathsf{rt} = \mathsf{Hash}_{\ell\text{-}\mathsf{SSB}}(\mathsf{hk}_{\ell\text{-}\mathsf{SSB}}, \pi)$. It sends rt to the verifier.
- **Second verifier's message:** $\mathcal{V}_{\mathsf{nsKilian}}$ computes a set of queries $(q_1, \ldots, q_\ell) \leftarrow \mathcal{Q}_{\mathsf{nsPCP}}(1^\kappa)$, and sends $(q_1, \ldots, q_\ell)$ to the prover.
- **Second prover's message:** $\mathcal{P}_{\mathsf{nsKilian}}$ computes for every $i \in [\ell]$ the opening $\mathsf{o}_i = \mathsf{Open}_{\ell\text{-}\mathsf{SSB}}(\mathsf{hk}_{\ell\text{-}\mathsf{SSB}}, \pi, q_i)$, and sends $\{\pi_{q_i}, \mathsf{o}_i\}_{i\in[\ell]}$ to the verifier.
- **Verdict:** $\mathcal{V}_{\mathsf{nsKilian}}$ accepts if and only if $\mathcal{V}_{\mathsf{nsPCP}}(x, (q_1, \ldots, q_\ell), (\pi_{q_1}, \ldots, \pi_{q_\ell})) = 1$ and for every $i \in [\ell]$, $\mathsf{Verify}_{\ell\text{-}\mathsf{SSB}}(\mathsf{hk}_{\ell\text{-}\mathsf{SSB}}, \mathsf{rt}, q_i, \pi_{q_i}, \mathsf{o}_i) = 1$.

Figure 4: The Protocol $(\mathcal{P}_{\mathsf{nsKilian}}, \mathcal{V}_{\mathsf{nsKilian}})$ for $\mathcal{L}$

With these ingredients, and setting κ to be $\Omega(n)^{1/\epsilon}$ such that $2^{\kappa^\epsilon} = \Omega$, the resulting Kilian's protocol is a 2^{κ^ϵ}-SSS argument w.r.t. the 2^{κ^ϵ}-hiding of the underlying SSB hash family, as we show below. Since $2^{\kappa^\epsilon} = \Omega(n)$, in an abuse of notation we say that the protocol is $\Omega(n)$-SSS.

Lemma 4.4. *Assuming that $\ell \leq \Omega$, $(\mathcal{P}_{\mathsf{nsKilian}}, \mathcal{V}_{\mathsf{nsKilian}})$ is a Ω-SSS interactive argument with respect to the 2^{κ^ϵ}-index hiding property of the underlying* SSB *hash family.*

[11] Note that the Ω'-index hiding property of the SSB is a non-interactive Ω'-decisional assumption.

Proof. For $(\mathsf{hk}_{\ell\text{-SSB}}, \mathsf{td}_{\ell\text{-SSB}}) \leftarrow \mathsf{Gen}_{\ell\text{-SSB}}(1^\kappa, L, Q)$, define $T(\mathsf{hk}_{\ell\text{-SSB}}) = Q$. We will show that $(\mathcal{P}_{\mathsf{nsKilian}}, \mathcal{V}_{\mathsf{nsKilian}})$ satisfies the properties in Definition 3.2. We will use the fact that $2^{\kappa^\epsilon} = 2^{((\log \Omega)^{1/\epsilon})^\epsilon} = \Omega$. In particular, since the ℓ-SSB hash family is 2^{κ^ϵ}-index-hiding by Lemma 4.2 (using that $\ell \leq \Omega = 2^{\kappa^\epsilon}$), it is in fact $\Omega(n)$-index hiding.

- **Ω-Somewhere Statistically Sound:** The Ω-somewhere statistically sound property of Definition 3.2[12] follows from the fact that for every $\mathrm{poly}(\Omega)$-size $\mathcal{P}^* = (\mathcal{P}_1^*, \mathcal{P}_2^*)$,

$$\Pr\left[\begin{array}{r|l} \exists\{a_j, \mathsf{o}_j\}_{j\in[\ell]} & Q \leftarrow \mathcal{Q}_{\mathsf{nsPCP}}(1^\kappa) \\ \text{s.t. } \mathcal{V}_{\mathsf{nsPCP}}(x, Q, (a_1, \ldots, a_\ell)) = 1 & (\mathsf{hk}_{\ell\text{-SSB}}, \mathsf{td}_{\ell\text{-SSB}}) \leftarrow \mathsf{Gen}_{\ell\text{-SSB}}(1^\kappa, L, Q) \\ \wedge\ \mathsf{Verify}_{\ell\text{-SSB}}(\mathsf{hk}_{\ell\text{-SSB}}, \mathsf{rt}, q_j, a_j, \mathsf{o}_j) = 1\ \forall j \in [\ell] & (\mathsf{rt}, \mathsf{state}) \leftarrow \mathcal{P}_1^*(x, \mathsf{hk}_{\ell\text{-SSB}}) \end{array}\right]$$

$$\leq \Pr\left[\begin{array}{l|r} \mathcal{V}_{\mathsf{nsPCP}}(x, Q, \mathsf{Invert}_{\ell\text{-SSB}}(\mathsf{td}_{\ell\text{-SSB}}, \mathsf{rt})) = 1 & Q \leftarrow \mathcal{Q}_{\mathsf{nsPCP}}(1^\kappa) \\ & (\mathsf{hk}_{\ell\text{-SSB}}, \mathsf{td}_{\ell\text{-SSB}}) \leftarrow \mathsf{Gen}_{\ell\text{-SSB}}(1^\kappa, L, Q) \\ & \mathsf{rt}, \mathsf{state} \leftarrow \mathcal{P}_1^*(x, \mathsf{hk}_{\ell\text{-SSB}}) \end{array}\right]$$

$$= \mathrm{negl}(\Omega),$$

  where the last equality follows from Corollary 4.3 and the fact that the ℓ-SSB hash family is 2^{κ^ϵ}-hiding (which is $\Omega(n)$-index hiding, as argued above). Furthermore, Corollary 4.3 gives that the reduction from the 2^{κ^ϵ}-hiding of the ℓ-SSB hash family to the Ω-somewhere statistical soundness is 2^{κ^ϵ}-straight-line.

- **Computational Indistinguishability:** In the formatted case, the pair $(\beta_1, T(\beta_1))$ is a pair $(\mathsf{hk}_{\ell\text{-SSB}}, Q)$ where $Q \leftarrow \mathcal{Q}_{\mathsf{nsPCP}}(1^\kappa)$ and $(\mathsf{hk}_{\ell\text{-SSB}}, \mathsf{td}_{\ell\text{-SSB}}) \leftarrow \mathsf{Gen}_{\ell\text{-SSB}}(1^\kappa, L, Q)$. Meanwhile, in the random case, the pair (β_1, β_2) is a pair $(\mathsf{hk}'_{\ell\text{-SSB}}, Q)$ where $Q, Q' \leftarrow \mathcal{Q}_{\mathsf{nsPCP}}(1^\kappa)$ and $(\mathsf{hk}'_{\ell\text{-SSB}}, \mathsf{td}'_{\ell\text{-SSB}}) \leftarrow \mathsf{Gen}_{\ell\text{-SSB}}(1^\kappa, L, Q')$. The indistinguishability of these two pairs for $\mathrm{poly}(\Omega)$-size distinguishers (and with probability non-negligible in Ω) follows from the $\Omega(n)$-index hiding property of the ℓ-SSB hash family via a 2^{κ^ϵ}-straight-line reduction: The reduction picks $Q \leftarrow \mathcal{Q}_{\mathsf{nsPCP}}(1^\kappa)$ at random. Then, to distinguish between $\mathsf{hk}_{\ell\text{-SSB}} \leftarrow \mathsf{Gen}_{\ell\text{-SSB}}(1^\kappa, L, Q)$ and $\mathsf{hk}_{\ell\text{-SSB}} \leftarrow \mathsf{Gen}_{\ell\text{-SSB}}(1^\kappa, L, Q')$ for an independent $Q' \leftarrow \mathcal{Q}_{\mathsf{nsPCP}}(1^\kappa)$, it feeds the pair $(Q, \mathsf{hk}_{\ell\text{-SSB}})$ to the distinguisher, and answers according to its response. Furthermore, by Theorem 4.2, the ℓ-SSB hash family is $\Omega(n)$-index hiding via a 2^{κ^ϵ}-straight-line reduction from the 2^{κ^ϵ}-index hiding of the underlying SSB hash family.

□

It follows from Theorem 3.3 that our instantiation of Kilian's protocol is 2^{κ^ϵ}-straight-line sound.

Theorem 4.5. *The protocol given in Figure 4 satisfies the following properties:*

- **Correctness:** *For any $x \in \mathcal{L}$ and $\epsilon > 0$,*

$$\Pr[(\mathcal{P}_{\mathsf{nsKilian}}, \mathcal{V}_{\mathsf{nsKilian}})(x) = 1] = 1.$$

[12] In our case, conditioned on the first three messages being $(\beta_1, \mathcal{P}^*(\beta_1), T(\beta_1))$, the remaining protocol is sound with probability 1.

- **Soundness:** *Assuming that the underlying* SSB *hash family is 2^{κ^ϵ}-hiding, the argument $(\mathcal{P}^*_{\mathsf{nsKilian}}, \mathcal{V}^*_{\mathsf{nsKilian}})$ for $\mathcal{L}$ is 2^{κ^ϵ}-straight-line sound. In particular, for any $x \notin \mathcal{L}$ and any $\mathrm{poly}(\Omega(n))$-size cheating prover $\mathcal{P}^*_{\mathsf{nsKilian}}$,*

$$\Pr[(\mathcal{P}^*_{\mathsf{nsKilian}}, \mathcal{V}_{\mathsf{nsKilian}})(x) = 1] = \mathrm{negl}(\Omega(n)).$$

Proof. Correctness is straightforward, and 2^{κ^ϵ}-straight-line soundness follows immediately from Theorem 3.3 and Lemma 4.4. □

Recall that the SSB hash family from Theorem 2.12 is sub-exponentially straight-line hiding assuming the sub-exponential hardness of LWE. Using this particular SSB hash family, and using the computational non-signaling PCPs given in Theorems 2.6, 2.7, and 2.8, we obtain the following corollaries:

Corollary 4.6. *For any $\mathrm{poly}(n) \le t \le \exp(n)$, assuming the sub-exponential hardness of* LWE*, there is $\epsilon > 0$ such that Kilian's protocol $(\mathcal{P}_{\mathsf{nsKilian}}, \mathcal{V}_{\mathsf{nsKilian}})$, instantiated with the t-computational non-signaling* PCP *for $\mathcal{L}_{\mathcal{U}}(t)$ from Theorem 2.6 and the ℓ-*SSB *hash family from Figure 4 with underlying* SSB *hash family given in Theorem 2.12, is 2^{κ^ϵ}-straight-line sound. In particular, assuming the sub-exponential quantum hardness of* LWE*, this protocol is post-quantum secure against size-$\mathrm{poly}(t)$ quantum provers, except with probability negligible in t.*

Furthermore, the prover runs in time $\mathrm{poly}(t)$, the verifier runs in time $n \cdot \mathrm{polylog}(t)$, and the communication complexity is $\mathrm{polylog}(t)$.

Proof. It remains to analyze the complexity of the protocol. The complexity claims follow from the following points:

- By Theorem 2.6, the size of the PCP proof is $\mathrm{poly}(t)$, so $\mathcal{P}_{\mathsf{nsKilian}}$ can compute the hash and openings in time $\mathrm{poly}(t)$.
- The size of a single SSB hash and opening is $\mathrm{poly}(\kappa) = \mathrm{polylog}(t)$, and the number of such SSB hashes and openings is $\ell = \kappa \cdot \mathrm{polylog}(t) = \mathrm{polylog}(t)$, for a total communication complexity of $\mathrm{polylog}(t)$.
- The verifier can check that all the answers and openings are consistent with rt in time $\mathrm{polylog}(t)$. He also runs $\mathcal{V}_{\mathsf{nsPCP}}$, which takes time $n \cdot \mathrm{poly}(\ell) = n \cdot \mathrm{polylog}(t)$, for a total verifier runtime of $n \cdot \mathrm{polylog}(t)$.

□

Corollary 4.7. *For any $\mathrm{poly}(n) \le t \le \exp(n)$ and $s = s(n) \ge \log t(n)$, assuming the sub-exponential hardness of* LWE*, there is $\epsilon > 0$ such that Kilian's protocol $(\mathcal{P}_{\mathsf{nsKilian}}, \mathcal{V}_{\mathsf{nsKilian}})$, instantiated with the 2^s-computational non-signaling* PCP *for $\mathsf{NL}_{\mathcal{U}}(t, s)$ from Theorem 2.7 and the ℓ-*SSB *hash family from Figure 4 with underlying* SSB *hash family given in Theorem 2.12, is 2^{κ^ϵ}-straight-line sound. In particular, assuming the sub-exponential quantum hardness of* LWE*, this protocol is post-quantum secure against size-$\mathrm{poly}(2^s)$ (and thus $\mathrm{poly}(t)$) quantum provers, except with probability negligible in 2^s.*

Furthermore, the honest prover runs in time $\mathrm{poly}(t)$*, the verifier runs in time* $n \cdot \mathrm{poly}(s)$*, and the communication complexity is* $\mathrm{poly}(s)$*.*

Proof. We analyze the complexity claims.

- By Theorem 2.7, the size of the PCP proof is $\mathrm{poly}(t)$, so $\mathcal{P}_{\mathsf{nsKilian}}$ can compute the hash and openings in time $\mathrm{poly}(t)$.
- The size of a single SSB hash and opening is $\mathrm{poly}(\kappa) = \mathrm{polylog}(2^s) = \mathrm{poly}(s)$, and the number of such SSB hashes and openings is $\ell = \kappa \cdot \mathrm{poly}(s) = \mathrm{poly}(s)$, for a total communication complexity of $\mathrm{poly}(s)$.
- The verifier can check that all the answers and openings are consistent with rt in time $\mathrm{poly}(s)$. He also runs $\mathcal{V}_{\mathsf{nsPCP}}$, which takes time $n \cdot \mathrm{poly}(\ell) = n \cdot \mathrm{poly}(s)$, for a total verifier runtime of $n \cdot \mathrm{poly}(s)$.

□

Finally, we have the following corollary for languages in BatchNP:

Corollary 4.8. *For any* NP *language* L*, batch size* $N(\cdot)$*, and instance size* m*, assuming the sub-exponential hardness of* LWE*, there is* $\epsilon > 0$ *such that Kilian's protocol* $(\mathcal{P}_{\mathsf{nsKilian}}, \mathcal{V}_{\mathsf{nsKilian}})$*, instantiated with the* $m \cdot N(m)$*-computational non-signaling* PCP *for* $\mathsf{L}^{\otimes N}$ *from Theorem 2.8 and the* ℓ*-*SSB *hash family from Figure 3 with underlying* SSB *hash family given in Theorem 2.12, is* 2^{κ^ϵ}*-straight-line sound. In particular, assuming the sub-exponential quantum hardness of* LWE*, this protocol is post-quantum secure against size-*$m \cdot N$ *quantum provers, except with probability negligible in* m *and* N*.*

Furthermore, the prover runs in time $\mathrm{poly}(m, N)$*, the verifier runs in time* $N \cdot \mathrm{poly}(m, \log N)$*, and the communication complexity is* $\mathrm{poly}(m, \log N)$*.*

Proof. The complexity claims follow from the following points:

- By Theorem 2.8, the size of the PCP proof is $\mathrm{poly}(m, N)$, so the prover can compute the hash and openings in time $\mathrm{poly}(m, N)$.
- The size of a single SSB hash and opening is $\mathrm{poly}(\kappa) = \mathrm{polylog}(m, \log N)$, and there are $\ell = \kappa \cdot \mathrm{poly}(m, \log N) = \mathrm{poly}(m, \log N)$ such hashes and openings, for a total communication complexity of $\mathrm{poly}(m, \log N)$.
- The verifier can check that all the answers and openings are consistent with rt in time $\mathrm{poly}(m, \log N)$. He also runs $\mathcal{V}_{\mathsf{nsPCP}}$, which takes time $N \cdot \mathrm{poly}(\ell) = N \cdot \mathrm{poly}(m, \log N)$.

□

5 SNARG for Languages with Non-Signaling PCPs

In this section, we construct SNARGs for languages with a non-signaling PCP, assuming the existence of a SNARG for BatchNP. This includes $\mathcal{L}_{\mathcal{U}}(t)$ for every $\mathrm{poly}(n) \le t \le \exp(n)$, and $\mathsf{N}\mathcal{L}_{\mathcal{U}}(t, s)$ for $\mathrm{poly}(n) \le t \le \exp(n)$ and $s = s(n) \ge \log t(n)$.

We begin by defining BatchNP and SNARGs for BatchNP.

5.1 BatchNP

For an NP relation R with corresponding language L, define

$$\mathsf{R}^{\otimes N} = \{((x_1, \ldots, x_N), (w_1, \ldots, w_N)) \ : \ (x_i, w_i) \in \mathsf{R} \ \forall i \in [N] \ \wedge \ |x_1| = \cdots = |x_N|\}$$

and

$$\mathsf{L}^{\otimes N} = \{(x_1, \ldots, x_N) \ : \ x_i \in \mathsf{L} \ \forall i \in [N] \ \wedge \ |x_1| = \cdots = |x_N|\}.$$

The class BatchNP consists of languages $\mathsf{L}^{\otimes N}$ for $\mathsf{L} \in \mathsf{NP}$.

5.1.1 SNARGs for BatchNP

Our SNARG for $\mathcal{L}$ relies on the existence of a SNARG for BatchNP, which we define below. We will be interested in the case where N is much larger than m, the size of a single instance x_i. We will consider two definitions. First, we consider a definition where the verifier is super-efficient (runs in time $\mathrm{poly}(m, \log N)$). Note that the size of a BatchNP instance is already $N \cdot m$, so in this case we will consider only BatchNP instances that have succinct descriptions. Second, we will consider a definition where the verifier is efficient (but not necessarily super-efficient), i.e. runs in time $\mathrm{poly}(m, N)$, but the communication is succinct (size $\mathrm{poly}(m, \log N)$). In this setting, the verifier reads the full instance.

To define SNARGs for BatchNP where the verifier is super-efficient, we first have to define succinct descriptions.

Definition 5.1. *(Succinct Description of a Tuple) A tuple $S \in (\{0,1\}^m)^N$ of size N has a* succinct description *if there exists a short string $\langle S \rangle \in \{0,1\}^{\mathrm{poly}(m, \log N)}$ and a uniform* PPT *Turing machine B that on input $\langle S \rangle$ and $i \in [N]$, outputs the i'th element of S.*

For notation, we let $B(\langle S \rangle)$ denote the set S, i.e. $B(\langle S \rangle) = \{B(\langle S \rangle, i)\}_{i \in [N]}$.

We next define SNARGs for BatchNP, both where the verifier reads the full BatchNP instance and where the instances have succinct descriptions.

Definition 5.2. *(SNARG for* BatchNP *(with Succinct Instances)) A* SNARG for a language $\mathsf{L}^{\otimes N} \in$ BatchNP with corresponding relation $\mathsf{R}^{\otimes N}$ (where the instance has a succinct description) *is a tuple of* PPT *algorithms $(\mathsf{Setup}_{\mathsf{L}^{\otimes N}}, \mathcal{P}_{\mathsf{L}^{\otimes N}}, \mathcal{V}_{\mathsf{L}^{\otimes N}})$ with the following syntax:*

- $\mathsf{Setup}_{\mathsf{L}^{\otimes N}}(1^\lambda, 1^m, N)$ *takes as input a security parameter λ and* NP *instance size m in unary, as well as a batch size N (in binary), and outputs a common reference string* crs.
- $\mathcal{P}_{\mathsf{L}^{\otimes N}}(\mathsf{crs}, X, W)$ *takes as input a* $\mathsf{crs} \in \{0,1\}^{\mathrm{poly}(\lambda, m, \log N)}$*, an instance $X = (x_1, \ldots, x_N) \in \{0,1\}^{N \times m}$, and a witness $W = (w_1, \ldots, w_N)$, and outputs a short proof $\sigma \in \{0,1\}^{\ell_{\mathsf{L}^{\otimes N}}}$, where $\ell_{\mathsf{L}^{\otimes N}} = \mathrm{poly}(\lambda, m, \log N)$.*
- $\mathcal{V}_{\mathsf{L}^{\otimes N}}(\mathsf{crs}, X, \sigma)$ *(resp. $\mathcal{V}_{\mathsf{L}^{\otimes N}}(\mathsf{crs}, \langle X \rangle, \sigma)$) takes as input the* $\mathsf{crs} \in \{0,1\}^{\mathrm{poly}(\lambda, m, \log N)}$*, $X = (x_1, \ldots, x_N) \in \{0,1\}^{N \times m}$ (resp. a short description $\langle X \rangle \in \{0,1\}^{\mathrm{poly}(\lambda, m, \log N)}$ of the instance X), and $\sigma \in \{0,1\}^{\ell_{\mathsf{L}^{\otimes N}}}$, and outputs 1 or 0 indicating accept or reject.*

These algorithms should satisfy the following completeness property:

If $(X, W) \in \mathsf{R}^{\otimes N}$*, then*

$$\Pr\left[\mathcal{V}_{\mathsf{L}^{\otimes N}}(\mathsf{crs}, X, \sigma) = 1 \text{ (resp. } \mathcal{V}_{\mathsf{L}^{\otimes N}}(\mathsf{crs}, \langle X \rangle, \sigma) = 1) \middle| \begin{array}{r} \mathsf{crs} \leftarrow \mathsf{Setup}_{\mathsf{L}^{\otimes N}}(1^\lambda, 1^m, N) \\ \sigma \leftarrow \mathcal{P}_{\mathsf{L}^{\otimes N}}(\mathsf{crs}, X, W) \end{array}\right] = 1.$$

Definition 5.3 (Σ-Soundness). *A SNARG* $(\mathsf{Setup}_{\mathsf{L}^{\otimes N}}, \mathcal{P}_{\mathsf{L}^{\otimes N}}, \mathcal{V}_{\mathsf{L}^{\otimes N}})$ *for* $\mathsf{L}^{\otimes N} \in \mathsf{BatchNP}$ *is said to be* Σ*-sound if for every cheating prover* $\mathcal{P}^*_{\mathsf{L}^{\otimes N}}$ *running in time* $\mathrm{poly}(\Sigma(\lambda, m, N))$*, there exists a negligible function* μ *such that for any* λ, m, N *and* $X \notin \mathsf{L}^{\otimes N}$ *where each instance is of size* m*,*

$$\Pr\left[\begin{array}{r} \mathcal{V}_{\mathsf{L}^{\otimes N}}(\mathsf{crs}, X, \sigma) = 1 \\ \text{(resp. } \mathcal{V}_{\mathsf{L}^{\otimes N}}(\mathsf{crs}, \langle X \rangle, \sigma) = 1) \end{array} \middle| \begin{array}{r} \mathsf{crs} \leftarrow \mathsf{Setup}_{\mathsf{L}^{\otimes N}}(1^\lambda, 1^m, N) \\ \sigma \leftarrow \mathcal{P}^*_{\mathsf{L}^{\otimes N}}(\mathsf{crs}) \end{array}\right] = \mathrm{negl}(\Sigma(\lambda, m, N)).$$

Theorem 5.4 ([CJJ21])**.** *Assuming the sub-exponential hardness of* LWE*, there is some* $\epsilon > 0$ *for which there exist* 2^{λ^ϵ}*-sound SNARGs for languages in* BatchNP *with succinct instances.*

5.2 SNARG for Languages with a Non-Signaling PCP

Suppose we have a Ω-computational non-signaling PCP $(\Pi, \mathcal{Q}_{\mathsf{nsPCP}}, \mathcal{V}_{\mathsf{nsPCP}})$ that is verifiable via tests (Definition 2.4) for a language $\mathcal{L}$. Let L be the size of the PCP and ℓ be the locality. Let N be the number of possible tests ζ (see Theorem 2.6), and let τ be the size of each test (where we pad tests that are not long enough), so that each test ζ can be written as $(\zeta_1, \ldots, \zeta_\tau)$ with $\zeta_i \in [L]$. Let $\mathcal{U}_{\mathsf{nsPCP}}$ be the Turing machine that checks each test, as in Definition 2.4.

At a high level, our SNARG for $\mathcal{L}$ works as follows: The honest prover first runs the BMW protocol on a computationally non-signaling PCP with SSB hash functions to produce a short commitment rt to the entire PCP. She then provides a short proof via the BatchNP SNARG that *all* possible verifier tests have accepting answers and openings. This final task is precisely a BatchNP statement: the claim that a given verifier test has accepting answers and openings is an NP statement, with witness the answers and openings; now the claim that *all possible* verifier tests have accepting answers and openings is a BatchNP statement.

We define the BatchNP language we will be concerned with, as well as the succinct description of the instances. Fix an ℓ-SSB hash family

$$(\mathsf{Gen}_{\ell\text{-}\mathsf{SSB}}, \mathsf{Hash}_{\ell\text{-}\mathsf{SSB}}, \mathsf{Open}_{\ell\text{-}\mathsf{SSB}}, \mathsf{Verify}_{\ell\text{-}\mathsf{SSB}}, \mathsf{Invert}_{\ell\text{-}\mathsf{SSB}})$$

(see Construction 3).

Let $\mathcal{R}$ be the NP relation where $(y, w) \in \mathcal{R}$ if

1. $y = (\zeta, x, \mathsf{hk}_{\ell\text{-}\mathsf{SSB}}, \mathsf{rt}) \in [L]^\tau \times \{0,1\}^n \times \{0,1\}^{\ell \cdot \ell_{\mathsf{hk}}} \times \{0,1\}^{\ell \cdot \ell_{\mathsf{hash}}}$;
2. $w = ((u_1, \ldots, u_\tau), (\mathsf{o}_1, \ldots, \mathsf{o}_\tau)) \in \{0,1\}^\tau \times \{0,1\}^{\tau \cdot \ell \cdot \ell_{\mathsf{o}}}$;
3. $\mathcal{U}_{\mathsf{nsPCP}}(x, \zeta, (u_1, \ldots, u_\tau)) = 1$; and
4. $\mathsf{Verify}_{\ell\text{-}\mathsf{SSB}}(\mathsf{hk}_{\ell\text{-}\mathsf{SSB}}, \mathsf{rt}, \zeta_i, u_i, \mathsf{o}_{\ell\text{-}\mathsf{SSB},i}) = 1 \ \forall i \in [\tau]$.

Let $\mathcal{M}$ be the corresponding language. Notice that the size of an instance is

$$m = \tau \cdot \log L + n + \ell \cdot (\ell_{\mathsf{hk}} + \ell_{\mathsf{hash}}). \tag{2}$$

We are interested in the BatchNP language $\mathcal{M}^{\otimes N}$.

Let B be a poly-time Turing machine that takes as input $\langle Y \rangle$, which is a succinct description of an element in $\mathcal{M}^{\otimes N}$, and an index $j \in [N]$, and outputs the j'th NP statement defined by $\langle Y \rangle$. More specifically, $\langle Y \rangle = (x, \mathsf{hk}_{\ell\text{-}\mathsf{SSB}}, \mathsf{rt})$, and $B(\langle Y \rangle, j) = (\zeta_j, \langle Y \rangle)$, where ζ_j is the j'th possible test (enumerating them in some order). We let Y denote the $\mathcal{M}^{\otimes N}$ instance corresponding to $\langle Y \rangle$.

5.2.1 SNARGs for $\mathcal{L}$ from SNARGs for BatchNP with Succinct Instances

We first construct SNARGs for $\mathcal{L}$ from SNARGs for BatchNP, assuming that the BatchNP SNARG verifier is super-efficient when the BatchNP instance admits a succinct description. This is indeed the case: our BatchNP instance is determined by the output of the hash on the PCP and thus can be described succinctly.

In what follows, let $(\mathsf{Setup}_{\mathcal{M}^{\otimes N}}, \mathcal{P}_{\mathcal{M}^{\otimes N}}, \mathcal{V}_{\mathcal{M}^{\otimes N}})$ be a SNARG for $\mathcal{M}^{\otimes N}$ with succinct instances, as in Definition 5.2.

Theorem 5.5. *The algorithms $(\mathsf{Setup}_{\mathcal{L}}, \mathcal{P}_{\mathcal{L}}, \mathcal{V}_{\mathcal{L}})$ defined in Figure 5 satisfy the following properties:*

- **Correctness:** *For every $x \in \mathcal{L}$,*

$$\Pr\left[\mathcal{V}_{\mathcal{L}}(\mathsf{crs}, x, \sigma) = 1 \;\middle|\; \begin{array}{r} \mathsf{crs} \leftarrow \mathsf{Setup}_{\mathcal{L}}(1^\kappa, 1^\lambda) \\ \sigma \leftarrow \mathcal{P}_{\mathcal{L}}(\mathsf{crs}, x) \end{array}\right] = 1.$$

- **Soundness:** *Assuming that*
  - *the underlying* SSB *hash family is 2^{κ^ϵ}-hiding,*
  - *the* PCP *is $\Omega = \Omega(n)$-computational non-signaling and is verified via tests, and that there are $N \leq \mathrm{poly}(\Omega)$ possible tests,*
  - *the* BatchNP *SNARG is Σ-sound, such that λ (defined in Figure 5) is $\leq \Omega$,*

  then for any $\mathrm{poly}(\Omega)$-size $\mathcal{P}^*$ and $x \notin \mathcal{L}$,

$$\Pr\left[\mathcal{V}_{\mathcal{L}}(\mathsf{crs}, x, \sigma)) = 1 \;\middle|\; \begin{array}{r} \mathsf{crs} \leftarrow \mathsf{Setup}_{\mathcal{L}}(1^\kappa, 1^\lambda) \\ \sigma \leftarrow \mathcal{P}_{\mathcal{L}}(\mathsf{crs}, x) \end{array}\right] = \mathrm{negl}(\Omega).$$

Proof. Correctness is straightforward. We now focus on proving soundness.

Suppose for the sake of contradiction that there is a $\mathrm{poly}(\Omega)$-size prover $\mathcal{P}^*$ and $x \notin \mathcal{L}$ for which there is non-negligible δ such that

$$\Pr\left[\mathcal{V}_{\mathcal{L}}(\mathsf{crs}, x, \sigma)) = 1 \;\middle|\; \begin{array}{r} \mathsf{crs} \leftarrow \mathsf{Setup}_{\mathcal{L}}(1^\kappa, 1^\lambda) \\ \sigma \leftarrow \mathcal{P}^*(\mathsf{crs}, x) \end{array}\right] = \delta(\Omega).$$

SNARG for $\mathcal{L}$ from SNARG for BatchNP with Succinct Instances

For $\epsilon > 0$ and $\Omega(\cdot)$, define $\kappa = (\log \Omega)^{1/\epsilon}$ and let λ be such that $\Sigma(\lambda, m, N) = 2^{\ell \cdot \ell_{\mathsf{hash}}}$.

- $\mathsf{Setup}_{\mathcal{L}}(1^\kappa, 1^\lambda)$ takes as input κ and λ in unary. It samples

$$Q = (q_1, \ldots, q_\ell) \leftarrow \mathcal{Q}_{\mathsf{nsPCP}}(1^\kappa), \text{ and } (\mathsf{hk}_{\ell\text{-SSB}}, \mathsf{td}_{\ell\text{-SSB}}) \leftarrow \mathsf{Gen}_{\ell\text{-SSB}}(1^\kappa, L, Q) \ .$$

  It also samples

$$\mathsf{crs}_{\mathcal{M}^{\otimes N}} \leftarrow \mathsf{Setup}_{\mathcal{M}^{\otimes N}}(1^\lambda, 1^m, N) \ ,$$

  and outputs $\mathsf{crs} = (\mathsf{hk}_{\ell\text{-SSB}}, \mathsf{crs}_{\mathcal{M}^{\otimes N}})$.

- $\mathcal{P}_{\mathcal{L}}$ takes as input the $\mathsf{crs} = (\mathsf{hk}_{\ell\text{-SSB}}, \mathsf{crs}_{\mathcal{M}^{\otimes N}})$ and an instance x. It computes

$$\pi \leftarrow \Pi(x) \text{ and } \mathsf{rt} = \mathsf{Hash}_{\ell\text{-SSB}}(\mathsf{hk}_{\ell\text{-SSB}}, \pi) \ .$$

  It then computes $\sigma_{\mathcal{M}^{\otimes N}} \leftarrow \mathcal{P}_{\mathcal{M}^{\otimes N}}(\mathsf{crs}_{\mathcal{M}^{\otimes N}}, Y, W)$, where

$$Y = \{(\zeta_j, x, \mathsf{hk}_{\ell\text{-SSB}}, \mathsf{rt})\}_{j \in [N]}$$

  (i.e. $\langle Y \rangle = (x, \mathsf{hk}_{\ell\text{-SSB}}, \mathsf{rt})$) and

$$W = \{((\pi_{\zeta_{j,1}}, \ldots, \pi_{\zeta_{j,\tau}}), (\mathsf{o}_{\zeta_{j,1}}, \ldots, \mathsf{o}_{\zeta_{j,\tau}}))\}_{j \in [N]} \ ,$$

  where $\mathsf{o}_q = \mathsf{Open}_{\ell\text{-SSB}}(\mathsf{hk}_{\ell\text{-SSB}}, \pi, q)$. It outputs $\sigma = (\mathsf{rt}, \sigma_{\mathcal{M}^{\otimes N}})$.

- $\mathcal{V}_{\mathcal{L}}$ takes as input $\mathsf{crs} = (\mathsf{hk}_{\ell\text{-SSB}}, \mathsf{crs}_{\mathcal{M}^{\otimes N}})$, instance x, and $\sigma = (\mathsf{rt}, \sigma_{\mathcal{M}^{\otimes N}})$. It runs and outputs the result of $\mathcal{V}_{\mathcal{M}^{\otimes N}}(\mathsf{crs}_{\mathcal{M}^{\otimes N}}, \langle Y \rangle, \sigma_{\mathcal{M}^{\otimes N}})$, where $\langle Y \rangle = (x, \mathsf{hk}_{\ell\text{-SSB}}, \mathsf{rt})$.

Figure 5: SNARG $(\mathsf{Setup}_{\mathcal{L}}, \mathcal{P}_{\mathcal{L}}, \mathcal{V}_{\mathcal{L}})(x)$ for $\mathcal{L}$

This is equal to

$$\begin{aligned}
\delta(\Omega) = \Pr\left[\begin{array}{r} \mathcal{V}_{\mathcal{L}}(\mathsf{crs}, x, \sigma)) = 1 \\ \wedge \ \mathcal{V}_{\mathsf{nsPCP}}(x, Q, \mathsf{Invert}_{\ell\text{-SSB}}(\mathsf{td}_{\ell\text{-SSB}}, \mathsf{rt})) = 1 \end{array} \middle| \begin{array}{r} \mathsf{crs} \leftarrow \mathsf{Setup}_{\mathcal{L}}(1^\kappa, 1^\lambda) \\ \sigma = (\mathsf{rt}, \sigma_{\mathcal{M}^{\otimes N}}) \leftarrow \mathcal{P}^*(\mathsf{crs}, x) \end{array}\right] \\
+ \Pr\left[\begin{array}{r} \mathcal{V}_{\mathcal{L}}(\mathsf{crs}, x, \sigma)) = 1 \\ \wedge \ \mathcal{V}_{\mathsf{nsPCP}}(x, Q, \mathsf{Invert}_{\ell\text{-SSB}}(\mathsf{td}_{\ell\text{-SSB}}, \mathsf{rt})) = 0 \end{array} \middle| \begin{array}{r} \mathsf{crs} \leftarrow \mathsf{Setup}_{\mathcal{L}}(1^\kappa, 1^\lambda) \\ \sigma = (\mathsf{rt}, \sigma_{\mathcal{M}^{\otimes N}}) \leftarrow \mathcal{P}^*(\mathsf{crs}, x) \end{array}\right] \\
\leq \Pr\left[\mathcal{V}_{\mathsf{nsPCP}}(x, Q, \mathsf{Invert}_{\ell\text{-SSB}}(\mathsf{td}_{\ell\text{-SSB}}, \mathsf{rt})) = 1 \middle| \begin{array}{r} \mathsf{crs} \leftarrow \mathsf{Setup}_{\mathcal{L}}(1^\kappa, 1^\lambda) \\ \sigma = (\mathsf{rt}, \sigma_{\mathcal{M}^{\otimes N}}) \leftarrow \mathcal{P}^*(\mathsf{crs}, x) \end{array}\right] \\
+ \Pr\left[\begin{array}{r} \mathcal{V}_{\mathcal{L}}(\mathsf{crs}, x, \sigma)) = 1 \\ \wedge \ \mathcal{V}_{\mathsf{nsPCP}}(x, Q, \mathsf{Invert}_{\ell\text{-SSB}}(\mathsf{td}_{\ell\text{-SSB}}, \mathsf{rt})) = 0 \end{array} \middle| \begin{array}{r} \mathsf{crs} \leftarrow \mathsf{Setup}_{\mathcal{L}}(1^\kappa, 1^\lambda) \\ \sigma = (\mathsf{rt}, \sigma_{\mathcal{M}^{\otimes N}}) \leftarrow \mathcal{P}^*(\mathsf{crs}, x) \end{array}\right] .
\end{aligned}$$

By Corollary 4.3 and the fact that a $2^{\kappa^\epsilon} = 2^{((\log \Omega)^{1/\epsilon})^\epsilon}$-hiding ℓ-SSB hash family is $\Omega(n)$-hiding, the first term above is $\mathrm{negl}(\Omega)$. In the above and what follows, Q denotes the ℓ locations the ℓ-SSB hash family are binding on (used to generate $\mathsf{hk}_{\ell\text{-SSB}}$), and $\mathsf{td}_{\ell\text{-SSB}}$ is the trapdoor generated alongside

$\mathsf{hk}_{\ell\text{-SSB}}$.

Therefore, the above implies that there exists $\delta'(\Omega) = \delta(\Omega) - \mathrm{negl}(\Omega)$ such that

$$\delta'(\Omega) \leq \Pr\left[\begin{array}{r|r} \mathcal{V}_{\mathcal{L}}(\mathsf{crs}, x, \sigma)) = 1 & \mathsf{crs} \leftarrow \mathsf{Setup}_{\mathcal{L}}(1^\kappa, 1^\lambda) \\ \wedge\ \mathcal{V}_{\mathsf{nsPCP}}(x, Q, \mathsf{Invert}_{\ell\text{-SSB}}(\mathsf{td}_{\ell\text{-SSB}}, \mathsf{rt})) = 0 & \sigma = (\mathsf{rt}, \sigma_{\mathcal{M}^{\otimes N}}) \leftarrow \mathcal{P}^*(\mathsf{crs}, x) \end{array}\right]$$
$$= \Pr\left[\begin{array}{r|r} \mathcal{V}_{\mathcal{M}^{\otimes N}}(\mathsf{crs}_{\mathcal{M}^{\otimes N}}, \langle Y\rangle, \sigma_{\mathcal{M}^{\otimes N}})) = 1 & \mathsf{crs} = (\mathsf{hk}_{\ell\text{-SSB}}, \mathsf{crs}_{\mathcal{M}^{\otimes N}}) \leftarrow \mathsf{Setup}_{\mathcal{L}}(1^\kappa, 1^\lambda) \\ \wedge\ Y \notin \mathcal{M}^{\otimes N} & \sigma = (\mathsf{rt}, \sigma_{\mathcal{M}^{\otimes N}}) \leftarrow \mathcal{P}^*(\mathsf{crs}, x) \end{array}\right],$$

where $\langle Y\rangle$ denotes $(x, \mathsf{hk}_{\ell\text{-SSB}}, \mathsf{rt})$, and the equality follows from the facts that $\mathcal{V}_{\mathcal{L}}$ simply runs $\mathcal{V}_{\mathcal{M}^{\otimes N}}$, and that $\mathcal{V}_{\mathsf{nsPCP}}(x, Q, \mathsf{Invert}_{\ell\text{-SSB}}(\mathsf{td}_{\ell\text{-SSB}}, \mathsf{rt})) = 0$ implies that $Y \notin \mathcal{M}^{\otimes N}$, since there is at least one test $\zeta \subseteq Q$ for which $\mathcal{U}_{\mathsf{nsPCP}}(\zeta, \mathsf{Invert}_{\ell\text{-SSB}}(\mathsf{td}_{\ell\text{-SSB}}, \mathsf{rt})\big|_\zeta) = 0$.

We will use $\mathcal{P}^*$ to break the Σ-security of the $\mathcal{M}^{\otimes N}$ SNARG as follows. By an averaging argument, there is some $\mathsf{hk}^*_{\ell\text{-SSB}}$ for which $\mathcal{P}^*(\mathsf{crs}, x)$ outputs $(\mathsf{rt}, \sigma_{\mathcal{M}^{\otimes N}})$ with $Y \notin \mathcal{M}^{\otimes N}$ and $\mathcal{V}_{\mathcal{M}^{\otimes N}}(\mathsf{crs}_{\mathcal{M}^{\otimes N}}\langle Y\rangle, \sigma_{\mathcal{M}^{\otimes N}}) = 1$ with probability $\geq \delta'(\Omega)$ conditioned on $\mathsf{crs} = (\mathsf{hk}^*_{\ell\text{-SSB}}, \mathsf{crs}_{\mathcal{M}^{\otimes N}})$ for some $\mathsf{crs}_{\mathcal{M}^{\otimes N}}$. Furthermore, there is some rt^* for which, with probability $\geq \frac{\delta'(\Omega)}{2^{\ell\cdot\ell_{\mathsf{hash}}}}$, this occurs and the rt output by $\mathcal{P}^*$ is equal to rt^*. In particular, for Y^* defined by $\langle Y^*\rangle = (x, \mathsf{hk}^*_{\ell\text{-SSB}}, \mathsf{rt}^*)$, we have that $Y^* \notin \mathcal{M}^{\otimes N}$.

$$\Pr\left[\begin{array}{r|r} \mathcal{V}_{\mathcal{M}^{\otimes N}}(\mathsf{crs}_{\mathcal{M}^{\otimes N}}, \langle Y^*\rangle, \sigma_{\mathcal{M}^{\otimes N}})) = 1 & \mathsf{crs}_{\mathcal{M}^{\otimes N}} \leftarrow \mathsf{Setup}_{\mathcal{M}^{\otimes N}}(1^\lambda, 1^m, N) \\ \wedge\ \mathsf{rt} = \mathsf{rt}^* & \mathsf{crs} := (\mathsf{hk}^*_{\ell\text{-SSB}}, \mathsf{crs}_{\mathcal{M}^{\otimes N}}) \\ & \sigma = (\mathsf{rt}, \sigma_{\mathcal{M}^{\otimes N}}) \leftarrow \mathcal{P}^*(\mathsf{crs}, x) \end{array}\right]$$
$$\geq \frac{\delta'(\Omega)}{2^{\ell\cdot\ell_{\mathsf{hash}}}} \geq \delta''(\Sigma(\lambda, m, N)),$$

where δ'' is a non-negligible function; such δ'' exists since we assumed that

$$\Sigma(\lambda, m, N) \geq 2^{\ell\cdot\ell_{\mathsf{hash}}} \geq 2^{\mathrm{poly}(\kappa)} = 2^{\mathrm{polylog}(\Omega)} \geq \Omega.$$

We next construct a cheating prover $\mathcal{P}^{**}$ for the $\mathcal{M}^{\otimes N}$ SNARG that breaks the Σ-soundness condition w.r.t. $Y^* \notin \mathcal{M}^{\otimes N}$, as follows. The cheating prover $\mathcal{P}^{**}$ takes as input $\mathsf{crs}_{\mathcal{M}^{\otimes N}} \leftarrow \mathsf{Setup}_{\mathcal{M}^{\otimes N}}(1^\kappa, 1^m, N)$, runs $\mathcal{P}^*$ on inputs $\mathsf{crs} = (\mathsf{hk}^*_{\ell\text{-SSB}}, \mathsf{crs}_{\mathcal{M}^{\otimes N}})$ and x, to get $(\mathsf{rt}, \sigma_{\mathcal{M}^{\otimes N}})$. When the Merkle root rt that $\mathcal{P}^*$ output is equal to rt^*, he outputs $\sigma_{\mathcal{M}^{\otimes N}}$, which fools $\mathcal{V}_{\mathcal{M}^{\otimes N}}$ with probability non-negligible in $\Sigma(\lambda, m, n)$. Furthermore, $\mathcal{P}^{**}$ runs in time $\mathrm{poly}(\Omega) \geq \mathrm{poly}(\lambda, m, N)$, since $N \leq \mathrm{poly}(\Omega)$ and $\lambda \leq \Omega$ by assumption. This contradicts the Σ-security of the $\mathcal{M}^{\otimes N}$ SNARG. □

Piecing together the following ingredients:

- a 2^{κ^ϵ}-hiding SSB hash family, which exists for some $\epsilon > 0$ assuming sub-exponential LWE (by Theorem 2.12),
- the t- or 2^s-computational non-signaling PCPs with $N = \mathrm{poly}(t)$ tests for $\mathcal{L}_{\mathcal{U}}(t)$ and $\mathsf{N}\mathcal{L}_{\mathcal{U}}(t, s)$ given in Theorems 2.6 and 2.7, respectively,
- the 2^{λ^ϵ}-secure SNARG for $\mathcal{M}^{\otimes N}$ given in Theorem 2.7 which exists for some $\epsilon > 0$ assuming sub-exponential LWE, which means we may take $\lambda = (\ell \cdot \ell_{\mathsf{hash}})^{1/\epsilon}$ (which equals $\mathrm{polylog}(t)$ and $\mathrm{poly}(s)$ in the case of $\mathcal{L}_{\mathcal{U}}(t)$ and $\mathsf{N}\mathcal{L}_{\mathcal{U}}(t, s)$) to satisfy $\Sigma(\lambda, m, N) = 2^{\lambda^\epsilon} = 2^{\ell\cdot\ell_{\mathsf{hash}}}$,

and taking $\epsilon > 0$ to be such that a 2^{κ^ϵ}-hiding SSB hash family and a 2^{λ^ϵ}-secure SNARG for $\mathcal{M}^{\otimes N}$ simultaneously exist assuming sub-exponential LWE, we have the following corollaries:

Corollary 5.6. *Let $t = t(n)$ be such that $\mathrm{poly}(n) \leq t(n) \leq \exp(n)$. Then, assuming sub-exponential* LWE*, there is a non-interactive argument for $\mathcal{L}_{\mathcal{U}}(t)$ that is sound except with probability $\mathrm{negl}(t)$ against $\mathrm{poly}(t)$-size cheating provers, where the honest prover runs in time $\mathrm{poly}(t)$, the verifier runs in time $\mathrm{poly}(n, \log t)$, and the communication complexity is $\mathrm{poly}(n, \log t)$.*

Proof. The SNARG for $\mathcal{L}_{\mathcal{U}}(t)$ is precisely that given in Figure 5 with the t-computational non-signaling PCP for $\mathcal{L}_{\mathcal{U}}(t)$ such that $N = \mathrm{poly}(t)$, which exists by Theorem 2.6, and setting $\epsilon > 0$ such that a 2^{κ^ϵ}-hiding SSB hash family and a 2^{κ^ϵ}-secure $\mathcal{M}^{\otimes N}$ SNARG exist assuming sub-exponential LWE. In this protocol, note that the prover first hashes the PCP, which takes time $\mathrm{poly}(t)$ (Theorem 2.6), and then emulates the prover from the $\mathcal{M}^{\otimes N}$ SNARG, which definitionally runs in time $\mathrm{poly}(\lambda, m, N) = \mathrm{poly}(t)$ (Definition 5.2). Note that $m = \tau \cdot \log L + n + \ell \cdot (\ell_{\mathsf{hk}} + \ell_{\mathsf{hash}}) = \mathrm{poly}(n, \kappa, \log L) = \mathrm{poly}(n, \log t)$. The proof string σ thus satisfies $|\sigma| = |\mathsf{rt}| + |\sigma_{\mathcal{M}^{\otimes N}}| = \mathrm{poly}(\kappa) + \mathrm{poly}(\lambda, m, \log N) = \mathrm{poly}(n, \log t)$. The verifier simply emulates $\mathcal{V}_{\mathcal{M}^{\otimes N}}$, which runs in time $\mathrm{poly}(\lambda, m, \log N) = \mathrm{poly}(n, \log t)$. □

Corollary 5.7. *Let $t = t(n)$ be such that $\mathrm{poly}(n) \leq t(n) \leq \exp(n)$ and let $s = s(n) \geq \log t(n)$. Assuming sub-exponential* LWE*, there is a non-interactive argument for $\mathsf{NL}_{\mathcal{U}}(t, s)$ that is sound except with probability $\mathrm{negl}(2^s)$ against $\mathrm{poly}(2^s)$-size cheating provers, where the honest prover runs in time $\mathrm{poly}(t)$, the verifier runs in time $\mathrm{poly}(n, s)$, and the communication complexity is $\mathrm{poly}(n, s)$.*

Proof. The SNARG for $\mathsf{NTISP}(t, s)$ is that given in Figure 5, instantiated with a 2^s-computational non-signaling PCP for $\mathsf{NTISP}(t, s)$ with $N = \mathrm{poly}(t)$ as given in Theorem 2.7, and $\epsilon > 0$ such that a 2^{κ^ϵ}-hiding SSB hash family a 2^{κ^ϵ}-secure $\mathcal{M}^{\otimes N}$ SNARG exist assuming sub-exponential LWE. We analyze the runtimes. First, the prover runs in time $\mathrm{poly}(t)$, since the PCP generated is of size $\mathrm{poly}(t)$, and the SNARG for $\mathcal{M}^{\otimes N}$ can also be generated in time $\mathrm{poly}(t)$. Since $m = \tau \cdot \log L + n + \ell \cdot (\ell_{\mathsf{hk}} + \ell_{\mathsf{hash}}) = \mathrm{poly}(n, \kappa, \log L) = \mathrm{poly}(n, s, \log t) = \mathrm{poly}(n, s)$, the proof string σ satisfies $|\sigma| = |\mathsf{rt}| + |\sigma_{\mathcal{M}^{\otimes N}}| = \mathrm{poly}(\kappa) + \mathrm{poly}(\lambda, m, \log N) = \mathrm{poly}(n, s, \log t) = \mathrm{poly}(n, s)$. Finally, the verifier emulates $\mathcal{V}_{\mathcal{M}^{\otimes N}}$, which runs in time $\mathrm{poly}(\lambda, m, \log N) = \mathrm{poly}(n, s, \log t) = \mathrm{poly}(n, s)$. □

5.2.2 SNARGs for $\mathcal{L}$ from SNARGs for BatchNP with Low Depth Verifier

In this section, we show that the assumption that the BatchNP SNARG verifier is super-efficient and takes as input succinct descriptions of BatchNP instances is not needed: in the case where the BatchNP SNARG verifier takes as input the full instance Y and runs in time polynomial in N, we can simply *delegate* these verifier checks back to the prover assuming that the checks are computable by a low depth circuit.

For this delegation of the verifier checks, we will use the SNARG for bounded depth computations constructed by [JKKZ21].

Theorem 5.8 ([JKKZ21]). *(SNARG for Size-S, Depth-D Circuits) Assuming the sub-exponential hardness of* LWE*, there is some $\epsilon > 0$ such that for any log-space uniform circuit C of size S and depth D, there are* PPT *algorithms $(\mathsf{Setup}_{\mathsf{JKKZ}}, \mathcal{P}_{\mathsf{JKKZ}}, \mathcal{V}_{\mathsf{JKKZ}})$ with syntax:*

- $\mathsf{Setup}_{\mathsf{JKKZ}}(1^\eta, S, 1^D)$ *takes as input a security parameter η in unary, the size S of the circuit in binary, and the depth D of the circuit in unary. It outputs a string* crs*.*
- $\mathcal{P}_{\mathsf{JKKZ}}(\mathsf{crs}, C, x)$ *takes as input the* crs*, circuit C of size S and depth D, and input x. She runs in time $\mathrm{poly}(\eta, S)$ and outputs a proof σ of size $D \cdot \mathrm{poly}(\eta, \log S)$.*
- $\mathcal{V}_{\mathsf{JKKZ}}(\mathsf{crs}, \langle C \rangle, x, \sigma)$ *takes as input the* crs*, a $\log S$ size description of the circuit C, the input x, and a short proof $\sigma \in \{0,1\}^{D \cdot \mathrm{poly}(\eta, \log S)}$. He runs in time $(D + |x|) \cdot \mathrm{poly}(\eta, \log S)$ and outputs either 0 or 1 indicating reject or accept.*

These algorithms satisfy the following properties:

- **Correctness:** *For C, x such that $C(x) = 1$,*

$$\Pr\left[\mathcal{V}_{\mathsf{JKKZ}}(\mathsf{crs}, \langle C \rangle, x, \sigma) = 1 \;\middle|\; \begin{array}{r} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\eta, S, 1^D) \\ \sigma \leftarrow \mathcal{P}_{\mathsf{JKKZ}}(\mathsf{crs}, C, x) \end{array}\right] = 1.$$

- **2^{η^ϵ}-Soundness:** *For C, x such that $C(x) \neq 1$, for any $\mathrm{poly}(2^{\eta^\epsilon})$-size $\mathcal{P}^*$, and for $\eta \geq \mathrm{polylog}(S)$,*

$$\Pr\left[\mathcal{V}_{\mathsf{JKKZ}}(\mathsf{crs}, \langle C \rangle, x, \sigma) = 1 \;\middle|\; \begin{array}{r} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\eta, S, 1^D) \\ \sigma \leftarrow \mathcal{P}^*(\mathsf{crs}) \end{array}\right] = \mathrm{negl}(2^{\eta^\epsilon}).$$

Fix a SNARG $(\mathsf{Setup}_{\mathcal{M}^{\otimes N}}, \mathcal{P}_{\mathcal{M}^{\otimes N}}, \mathcal{V}_{\mathcal{M}^{\otimes N}})$ for $\mathcal{M}^{\otimes N}$ as in Definition 5.2, where $\mathcal{V}_{\mathcal{M}^{\otimes N}}$ takes as input the full instance X rather than just a description. Suppose that the circuit $\mathcal{V}_{\mathcal{M}^{\otimes N}}$ has size $S = \mathrm{poly}(\lambda, m, N)$ and depth D. Let $\mathcal{V}'_{\mathcal{M}^{\otimes N}}$ denote the algorithm that takes as input $(\mathsf{crs}_{\mathcal{M}^{\otimes N}}, \langle Y \rangle, \sigma_{\mathcal{M}^{\otimes N}})$, computes $Y = B(\langle Y \rangle)$, and then runs $\mathcal{V}_{\mathcal{M}^{\otimes N}}(\mathsf{crs}_{\mathcal{M}^{\otimes N}}, Y, \sigma_{\mathcal{M}^{\otimes N}})$. Denote by $S(B)$ and $D(B)$ the size and depth respectively of a circuit computing $B(\cdot, \cdot)$, as defined in Definition 5.1. Note that the circuit computing $\mathcal{V}'_{\mathcal{M}^{\otimes N}}$ has size $S' = S + N \cdot S(B) = S + N \cdot \mathrm{poly}(m, \log N)$ and depth $D' = D + D(B) = D + \mathrm{poly}(m, \log N)$. Let $(\mathsf{Setup}_{\mathsf{JKKZ}}, \mathcal{P}_{\mathsf{JKKZ}}, \mathcal{V}_{\mathsf{JKKZ}})$ be the SNARG for circuits of size S' and depth D' given in Theorem 5.8.

Our SNARG for $\mathcal{L}$ is described in Figure 6.

Theorem 5.9. *The algorithms $(\mathsf{Setup}_{\mathcal{L}}, \mathcal{P}_{\mathcal{L}}, \mathcal{V}_{\mathcal{L}})$ defined in Figure 6 satisfy the following properties:*

- **Correctness:** *For every $x \in \mathcal{L}$,*

$$\Pr\left[\mathcal{V}_{\mathcal{L}}(\mathsf{crs}, x, \sigma) = 1 \;\middle|\; \begin{array}{r} \mathsf{crs} \leftarrow \mathsf{Setup}_{\mathcal{L}}(1^\kappa, 1^\lambda) \\ \sigma \leftarrow \mathcal{P}_{\mathcal{L}}(\mathsf{crs}, x) \end{array}\right] = 1.$$

- **Soundness:** *Assuming that:*

SNARG for $\mathcal{L}$ from SNARG for BatchNP

For $\epsilon > 0$, define $\kappa = (\log \Omega)^{1/\epsilon}$ and let λ be such that $\Sigma(\lambda, m, N) \geq 2^{\ell \cdot \ell_{\mathsf{hash}}}$. Let $\eta = (\ell \cdot \ell_{\mathsf{hash}} + \ell_{\mathcal{M}^{\otimes N}})^{1/\epsilon}$.

- $\mathsf{Setup}_{\mathcal{L}}(1^\kappa, 1^\lambda)$ takes as input κ and λ in unary. It samples

$$Q = (q_1, \ldots, q_\ell) \leftarrow \mathcal{Q}_{\mathsf{nsPCP}}(1^\kappa), \text{ and } (\mathsf{hk}_{\ell\text{-SSB}}, \mathsf{td}_{\ell\text{-SSB}}) \leftarrow \mathsf{Gen}_{\ell\text{-SSB}}(1^\kappa, L, Q).$$

  It also samples

$$\mathsf{crs}_{\mathcal{M}^{\otimes N}} \leftarrow \mathsf{Setup}_{\mathcal{M}^{\otimes N}}(1^\lambda, 1^m, N)$$

  and

$$\mathsf{crs}_{\mathsf{JKKZ}} \leftarrow \mathsf{Setup}_{\mathsf{JKKZ}}(1^\eta, S, 1^D),$$

  and outputs $\mathsf{crs} = (\mathsf{hk}_{\ell\text{-SSB}}, \mathsf{crs}_{\mathcal{M}^{\otimes N}}, \mathsf{crs}_{\mathsf{JKKZ}})$.

- $\mathcal{P}_{\mathcal{L}}$ takes as input the $\mathsf{crs} = (\mathsf{hk}_{\ell\text{-SSB}}, \mathsf{crs}_{\mathcal{M}^{\otimes N}}, \mathsf{crs}_{\mathsf{JKKZ}})$ and an instance x. It computes

$$\pi \leftarrow \Pi(x) \text{ and } \mathsf{rt} = \mathsf{Hash}_{\ell\text{-SSB}}(\mathsf{hk}_{\ell\text{-SSB}}, \pi) \ .$$

  It then computes $\sigma_{\mathcal{M}^{\otimes N}} \leftarrow \mathcal{P}_{\mathcal{M}^{\otimes N}}(\mathsf{crs}_{\mathcal{M}^{\otimes N}}, Y, W)$, where

$$Y = \{(\zeta_j, x, \mathsf{hk}_{\ell\text{-SSB}}, \mathsf{rt})\}_{j \in [N]}$$

  and

$$W = \{((\pi_{\zeta_{j,1}}, \ldots, \pi_{\zeta_{j,\tau}}), (\mathsf{o}_{\zeta_{j,1}}, \ldots, \mathsf{o}_{\zeta_{j,\tau}}))\}_{j \in [N]} \ ,$$

  where $\mathsf{o}_q = \mathsf{Open}_{\ell\text{-SSB}}(\mathsf{hk}_{\ell\text{-SSB}}, \pi, q)$. Finally, it computes

$$\sigma_{\mathsf{JKKZ}} \leftarrow \mathcal{P}_{\mathsf{JKKZ}}(\mathsf{crs}_{\mathsf{JKKZ}}, \mathcal{V}'_{\mathcal{M}^{\otimes N}}, (\mathsf{crs}_{\mathcal{M}^{\otimes N}}, \langle Y \rangle, \sigma_{\mathcal{M}^{\otimes N}})).$$

  It outputs $\sigma = (\mathsf{rt}, \sigma_{\mathcal{M}^{\otimes N}}, \sigma_{\mathsf{JKKZ}})$.

- $\mathcal{V}_{\mathcal{L}}$ takes as input $\mathsf{crs} = (\mathsf{hk}_{\ell\text{-SSB}}, \mathsf{crs}_{\mathcal{M}^{\otimes N}}, \mathsf{crs}_{\mathsf{JKKZ}})$, instance x, and $\sigma = (\mathsf{rt}, \sigma_{\mathcal{M}^{\otimes N}}, \sigma_{\mathsf{JKKZ}})$. It runs and outputs the result of $\mathcal{V}_{\mathsf{JKKZ}}(\mathsf{crs}_{\mathsf{JKKZ}}, \langle \mathcal{V}'_{\mathcal{M}^{\otimes N}} \rangle, (\mathsf{crs}_{\mathcal{M}^{\otimes N}}, \langle Y \rangle, \sigma_{\mathcal{M}^{\otimes N}}), \sigma_{\mathsf{JKKZ}})$, where $\langle Y \rangle = (x, \mathsf{hk}_{\ell\text{-SSB}}, \mathsf{rt})$.

Figure 6: SNARG $(\mathsf{Setup}_{\mathcal{L}}, \mathcal{P}_{\mathcal{L}}, \mathcal{V}_{\mathcal{L}})(x)$ for $\mathcal{L}$

  - *the underlying* SSB *hash family is* 2^{κ^ϵ}*-hiding,*
  - *the* PCP *is* Ω*-computational non-signaling and verified via tests, of which there are* $N \leq \mathrm{poly}(\Omega)$*,*
  - *the* $\mathcal{M}^{\otimes N}$ *SNARG is* Σ*-sound, such that* λ *(defined in Figure 6) is* $\leq \Omega$*,*
  - $\mathcal{V}_{\mathcal{M}^{\otimes N}}$ *is a log-space uniform circuit of depth* D*,*
  - $(\mathsf{Setup}_{\mathsf{JKKZ}}, \mathcal{P}_{\mathsf{JKKZ}}, \mathcal{V}_{\mathsf{JKKZ}})$ *has* 2^{η^ϵ}*-soundness,*

then for any $\mathrm{poly}(\Omega)$*-size* $\mathcal{P}^*$ *and* $x \notin \mathcal{L}$*,*

$$\Pr\left[\mathcal{V}_{\mathcal{L}}(\mathsf{crs}, x, \sigma)) = 1 \,\middle|\, \begin{matrix}\mathsf{crs} \leftarrow \mathsf{Setup}_{\mathcal{L}}(1^\kappa, 1^\lambda) \\ \sigma \leftarrow \mathcal{P}_{\mathcal{L}}(\mathsf{crs}, x)\end{matrix}\right] = \mathrm{negl}(\Omega).$$

Proof. Correctness is straightforward. We focus on proving soundness. Suppose for the sake of contradiction that there is a $\mathrm{poly}(\Omega)$-size prover $\mathcal{P}^*$ and $x \notin \mathcal{L}$ and a non-negligible δ such that

$$\Pr\left[\mathcal{V}_{\mathcal{L}}(\mathsf{crs}, x, \sigma)) = 1 \,\middle|\, \begin{matrix}\mathsf{crs} \leftarrow \mathsf{Setup}_{\mathcal{L}}(1^\kappa, 1^\lambda) \\ \sigma \leftarrow \mathcal{P}^*(\mathsf{crs}, x)\end{matrix}\right] = \delta(\Omega).$$

As in the proof of Theorem 5.5, this implies that there is non-negligible δ' such that

$$\delta'(\Omega) \leq \Pr\left[\begin{matrix}\mathcal{V}_{\mathcal{L}}(\mathsf{crs}, x, \sigma)) = 1 \\ \wedge\ \mathcal{V}_{\mathsf{nsPCP}}(x, Q, \mathsf{Invert}_{\ell\text{-}\mathsf{SSB}}(\mathsf{td}_{\ell\text{-}\mathsf{SSB}}, \mathsf{rt})) = 0\end{matrix}\,\middle|\, \begin{matrix}\mathsf{crs} \leftarrow \mathsf{Setup}_{\mathcal{L}}(1^\kappa, 1^\lambda) \\ \sigma = (\mathsf{rt}, \sigma_{\mathcal{M}^{\otimes N}}, \sigma_{\mathsf{JKKZ}}) \leftarrow \mathcal{P}^*(\mathsf{crs}, x)\end{matrix}\right].$$

Using the fact that $\mathcal{V}_{\mathcal{L}}$ simply runs $\mathcal{V}_{\mathsf{JKKZ}}$, and that $\mathcal{V}_{\mathsf{nsPCP}}(x, Q, \mathsf{Invert}_{\ell\text{-}\mathsf{SSB}}(\mathsf{td}_{\ell\text{-}\mathsf{SSB}}, \mathsf{rt})) = 0$ implies that $Y \notin \mathcal{M}^{\otimes N}$, since there is at least one test $\zeta \subseteq Q$ for which $\mathcal{U}_{\mathsf{nsPCP}}(\zeta, \mathsf{Invert}_{\ell\text{-}\mathsf{SSB}}(\mathsf{td}_{\ell\text{-}\mathsf{SSB}}, \mathsf{rt})\big|_\zeta) = 0$, we obtain that this is equal to

$$= \Pr\left[\begin{matrix}\mathcal{V}_{\mathsf{JKKZ}}\left(\begin{matrix}\mathsf{crs}_{\mathsf{JKKZ}}, \mathcal{V}'_{\mathcal{M}^{\otimes N}}, \\ (\mathsf{crs}_{\mathcal{M}^{\otimes N}}, \langle Y\rangle, \sigma_{\mathcal{M}^{\otimes N}}), \\ \sigma_{\mathsf{JKKZ}}\end{matrix}\right) = 1 \\ \wedge\ Y \notin \mathcal{M}^{\otimes N}\end{matrix}\,\middle|\, \begin{matrix}\mathsf{crs} = (\mathsf{hk}_{\ell\text{-}\mathsf{SSB}}, \mathsf{crs}_{\mathcal{M}^{\otimes N}}, \mathsf{crs}_{\mathsf{JKKZ}}) \leftarrow \mathsf{Setup}_{\mathcal{L}}(1^\kappa, 1^\lambda) \\ \sigma = (\mathsf{rt}, \sigma_{\mathcal{M}^{\otimes N}}, \sigma_{\mathsf{JKKZ}}) \leftarrow \mathcal{P}^*(\mathsf{crs}, x)\end{matrix}\right],$$

where $\langle Y\rangle$ denotes $(x, \mathsf{hk}_{\ell\text{-}\mathsf{SSB}}, \mathsf{rt})$. We can split this into whether $\sigma_{\mathcal{M}^{\otimes N}}$ is an accepting proof for $Y \in \mathcal{M}^{\otimes N}$:

$$\leq \Pr\left[\begin{matrix}\mathcal{V}_{\mathsf{JKKZ}}\left(\begin{matrix}\mathsf{crs}_{\mathsf{JKKZ}}, \mathcal{V}'_{\mathcal{M}^{\otimes N}}, \\ (\mathsf{crs}_{\mathcal{M}^{\otimes N}}, \langle Y\rangle, \sigma_{\mathcal{M}^{\otimes N}}), \\ \sigma_{\mathsf{JKKZ}}\end{matrix}\right) = 1 \\ \wedge\ \mathcal{V}'_{\mathcal{M}^{\otimes N}}(\mathsf{crs}_{\mathcal{M}^{\otimes N}}, \langle Y\rangle, \sigma_{\mathcal{M}^{\otimes N}}) = 0\end{matrix}\,\middle|\, \begin{matrix}\mathsf{crs} = (\mathsf{hk}_{\ell\text{-}\mathsf{SSB}}, \mathsf{crs}_{\mathcal{M}^{\otimes N}}, \mathsf{crs}_{\mathsf{JKKZ}}) \leftarrow \mathsf{Setup}_{\mathcal{L}}(1^\kappa, 1^\lambda) \\ \sigma = (\mathsf{rt}, \sigma_{\mathcal{M}^{\otimes N}}, \sigma_{\mathsf{JKKZ}}) \leftarrow \mathcal{P}^*(\mathsf{crs}, x)\end{matrix}\right] \tag{3}$$

$$+ \Pr\left[\begin{matrix}\mathcal{V}'_{\mathcal{M}^{\otimes N}}(\mathsf{crs}_{\mathcal{M}^{\otimes N}}, \langle Y\rangle, \sigma_{\mathcal{M}^{\otimes N}}) = 1 \\ \wedge\ Y \notin \mathcal{M}^{\otimes N}\end{matrix}\,\middle|\, \begin{matrix}\mathsf{crs} = (\mathsf{hk}_{\ell\text{-}\mathsf{SSB}}, \mathsf{crs}_{\mathcal{M}^{\otimes N}}, \mathsf{crs}_{\mathsf{JKKZ}}) \leftarrow \mathsf{Setup}_{\mathcal{L}}(1^\kappa, 1^\lambda) \\ \sigma = (\mathsf{rt}, \sigma_{\mathcal{M}^{\otimes N}}, \sigma_{\mathsf{JKKZ}}) \leftarrow \mathcal{P}^*(\mathsf{crs}, x)\end{matrix}\right]. \tag{4}$$

To get a contradiction we argue that both terms of this sum are negligible in Ω. The term (4) is $\mathrm{negl}(\Omega)$ for precisely the same reason as in the proof of Theorem 5.5: we can otherwise break the Σ-soundness of the $\mathcal{M}^{\otimes N}$ SNARG by finding an accepting proof $\sigma_{\mathcal{M}^{\otimes N}}$ for some instance $Y^* = (x, \mathsf{hk}^*_{\ell\text{-}\mathsf{SSB}}, \mathsf{rt}^*_{\ell\text{-}\mathsf{SSB}})$ with probability $\frac{\Delta(\Omega)}{2^{\ell \cdot \ell_{\mathsf{hash}}}} = \Delta'(\Sigma(\lambda, m, N))$, where Δ and Δ' are non-negligible functions.

As for the term (3), suppose that it is equal to $\Delta(\Omega)$, for some non-negligible Δ. By an averaging argument, there is some $\mathsf{hk}^*_{\ell\text{-}\mathsf{SSB}}$ and $\mathsf{crs}^*_{\mathcal{M}^{\otimes N}}$ for which (3) $\geq \Delta(\Omega)$, conditioned on $\mathsf{crs} =$

$(\mathsf{hk}^*_{\ell\text{-SSB}}, \mathsf{crs}^*_{\mathcal{M}^{\otimes N}}, \mathsf{crs}_{\mathsf{JKKZ}})$. Next, conditioning on crs being of the form $(\mathsf{hk}^*_{\ell\text{-SSB}}, \mathsf{crs}^*_{\mathcal{M}^{\otimes N}}, \mathsf{crs}_{\mathsf{JKKZ}})$, there is some $\mathsf{rt}^*, \sigma^*_{\mathcal{M}^{\otimes N}}$ for which, with probability $\geq \frac{\Delta(\Omega)}{2^{\ell \cdot \ell_{\mathsf{hash}} + \ell_{\mathcal{M}^{\otimes N}}}}$, the conditions of (3) hold *and* $\mathcal{P}^*$ outputs $\sigma = (\mathsf{rt}, \sigma_{\mathcal{M}^{\otimes N}}, \sigma_{\mathsf{JKKZ}})$ and $\mathsf{rt} = \mathsf{rt}^*, \sigma_{\mathcal{M}^{\otimes N}} = \sigma^*_{\mathcal{M}^{\otimes N}}$. That is, letting $\langle Y^* \rangle = (x, \mathsf{hk}^*_{\ell\text{-SSB}}, \mathsf{rt}^*)$,

$$\Pr\left[\begin{array}{r} \mathcal{V}_{\mathsf{JKKZ}}\left(\begin{array}{r}\mathsf{crs}_{\mathsf{JKKZ}}, \mathcal{V}'_{\mathcal{M}^{\otimes N}},\\ (\mathsf{crs}^*_{\mathcal{M}^{\otimes N}}, \langle Y^* \rangle, \sigma^*_{\mathcal{M}^{\otimes N}}),\\ \sigma_{\mathsf{JKKZ}}\end{array}\right) = 1 \\ \wedge\ \mathcal{V}'_{\mathcal{M}^{\otimes N}}(\mathsf{crs}^*_{\mathcal{M}^{\otimes N}}, \langle Y^* \rangle, \sigma^*_{\mathcal{M}^{\otimes N}}) = 0 \\ \wedge\ \mathsf{rt} = \mathsf{rt}^* \\ \wedge\ \sigma_{\mathcal{M}^{\otimes N}} = \sigma^*_{\mathcal{M}^{\otimes N}}\end{array} \middle| \begin{array}{r}\\ \mathsf{crs}_{\mathsf{JKKZ}} \leftarrow \mathsf{Setup}_{\mathsf{JKKZ}}(1^\eta, S, 1^D) \\ \mathsf{crs} := (\mathsf{hk}^*_{\ell\text{-SSB}}, \mathsf{crs}^*_{\mathcal{M}^{\otimes N}}, \mathsf{crs}_{\mathsf{JKKZ}}) \\ \sigma = (\mathsf{rt}, \sigma_{\mathcal{M}^{\otimes N}}, \sigma_{\mathsf{JKKZ}}) \leftarrow \mathcal{P}^*(\mathsf{crs}, x)\end{array}\right] \geq \frac{\Delta(\Omega)}{2^{\ell \cdot \ell_{\mathsf{hash}} + \ell_{\mathcal{M}^{\otimes N}}}},$$

where $\ell_{\mathcal{M}^{\otimes N}} = \mathrm{poly}(\lambda, m, \log N)$ is the length of $\sigma_{\mathcal{M}^{\otimes N}}$.

Letting Z^* denote $(\mathsf{crs}^*_{\mathcal{M}^{\otimes N}}, \langle Y^* \rangle, \sigma^*_{\mathcal{M}^{\otimes N}})$, so that $\mathcal{V}'_{\mathcal{M}^{\otimes N}}(Z^*) = 0$, this implies that

$$\begin{aligned}
&\Pr\left[\mathcal{V}_{\mathsf{JKKZ}}\left(\mathsf{crs}_{\mathsf{JKKZ}}, \mathcal{V}'_{\mathcal{M}^{\otimes N}}, Z^*, \sigma_{\mathsf{JKKZ}}\right) = 1 \;\middle|\; \begin{array}{r}\mathsf{crs} = (\mathsf{hk}_{\ell\text{-SSB}}, \mathsf{crs}_{\mathcal{M}^{\otimes N}}, \mathsf{crs}_{\mathsf{JKKZ}}) \leftarrow \mathsf{Setup}_{\mathcal{L}}(1^\kappa, 1^\lambda)\\ \sigma = (\mathsf{rt}, \sigma_{\mathcal{M}^{\otimes N}}, \sigma_{\mathsf{JKKZ}}) \leftarrow \mathcal{P}^*(\mathsf{crs}, x)\end{array}\right] \\
\geq &\Pr\left[\begin{array}{r}\mathcal{V}_{\mathsf{JKKZ}}\left(\mathsf{crs}_{\mathsf{JKKZ}}, \mathcal{V}'_{\mathcal{M}^{\otimes N}}, Z^*, \sigma_{\mathsf{JKKZ}}\right) = 1\\ \wedge\ \mathsf{rt} = \mathsf{rt}^*\\ \wedge\ \sigma_{\mathcal{M}^{\otimes N}} = \sigma^*_{\mathcal{M}^{\otimes N}}\end{array} \middle| \begin{array}{r}\mathsf{crs} = (\mathsf{hk}_{\ell\text{-SSB}}, \mathsf{crs}_{\mathcal{M}^{\otimes N}}, \mathsf{crs}_{\mathsf{JKKZ}}) \leftarrow \mathsf{Setup}_{\mathcal{L}}(1^\kappa, 1^\lambda)\\ \sigma = (\mathsf{rt}, \sigma_{\mathcal{M}^{\otimes N}}, \sigma_{\mathsf{JKKZ}}) \leftarrow \mathcal{P}^*(\mathsf{crs}, x)\end{array}\right] \\
&\geq \frac{\Delta(\Omega)}{2^{\ell \cdot \ell_{\mathsf{hash}} + \ell_{\mathcal{M}^{\otimes N}}}} \\
&= \Delta'(2^{\eta^\epsilon}),
\end{aligned}$$

where Δ' is a non-negligible function (it exists because η is defined such that $2^{\eta^\epsilon} = 2^{\ell \cdot \ell_{\mathsf{hash}} + \ell_{\mathcal{M}^{\otimes N}}} > \Omega$). This then contradicts the 2^{η^ϵ}-soundness of the JKKZ SNARG, as $\mathcal{P}^*(\mathsf{crs}, x)$ outputs σ_{JKKZ} for which $\mathcal{V}_{\mathsf{JKKZ}}(\mathsf{crs}_{\mathsf{JKKZ}}, \mathcal{V}'_{\mathcal{M}^{\otimes N}}, Z^*, \sigma_{\mathsf{JKKZ}}) = 1$ with probability non-negligible in 2^{η^ϵ}.

□

Assuming sub-exponential LWE, there is some $\epsilon > 0$ such that both the following hold: a 2^{κ^ϵ}-hiding SSB hash family exists and $(\mathsf{Setup}_{\mathsf{JKKZ}}, \mathcal{P}_{\mathsf{JKKZ}}, \mathcal{V}_{\mathsf{JKKZ}})$ has 2^{η^ϵ}-soundness. Assuming this, and using the computational non-signaling PCPs for $\mathcal{L}_{\mathcal{U}}(t)$ and $\mathsf{N}\mathcal{L}_{\mathcal{U}}(t, s)$ from Theorems 2.6 and 2.7, we get the following corollaries:

Corollary 5.10. *For any* $\mathrm{poly}(n) \leq t(n) \leq \exp(n)$*, there exists a non-interactive argument for* $\mathcal{L}_{\mathcal{U}}(t)$ *that is sound against* $\mathrm{poly}(t)$*-size cheating provers, assuming that*

- LWE *is sub-exponentially hard,*
- *there is a* Σ*-sound SNARG for* $\mathcal{M}^{\otimes N}$*,*
- *the* $\mathcal{M}^{\otimes N}$ *verifier* $\mathcal{V}_{\mathcal{M}^{\otimes N}}$ *is a log-space uniform circuit of depth* D*.*

The prover runs in time $\mathrm{poly}(t)$*, the verifier runs in time* $D \cdot \mathrm{poly}(n, \lambda, \log t)$*, and the communication complexity is* $D \cdot \mathrm{poly}(n, \lambda, \log t)$*.*

Proof. The non-interactive argument for $\mathcal{L}_{\mathcal{U}}(t)$ is that given in Figure 5.9, with ϵ corresponding to the LWE assumption, and a t-computational non-signaling PCP for $\mathcal{L}_{\mathcal{U}}(t)$ that is verified via $N = \mathrm{poly}(t)$ tests, as given in Theorem 2.6.

To analyze the runtimes, we first compute some sizes:

- $\ell \cdot \ell_{\mathsf{hash}} = \kappa \cdot \mathrm{polylog}(t) \cdot \mathrm{poly}(\kappa) = \mathrm{polylog}(t)$,
- $\ell_{\mathcal{M}^{\otimes N}} = \mathrm{poly}(\lambda, m, \log N) = \mathrm{poly}(n, \lambda, \log t)$,
- $S = \mathrm{poly}(t)$ since $\mathcal{V}_{\mathcal{M}^{\otimes N}}$ runs in time $S = \mathrm{poly}(\lambda, m, N) = \mathrm{poly}(t)$.

Thus, $\eta = (\ell \cdot \ell_{\mathsf{hash}} + \ell_{\mathcal{M}^{\otimes N}})^{1/\epsilon} = \mathrm{poly}(n, \lambda, \log t)$. Now, we analyze the runtimes and complexities.

- The prover has to do three things: compute the PCP π, which takes time $\mathrm{poly}(t)$; compute Y and W, and compute $\sigma_{\mathcal{M}^{\otimes N}}$, which together takes time $\mathrm{poly}(\lambda, m, N) = \mathrm{poly}(t)$; and finally compute σ_{JKKZ}, which takes time $\mathrm{poly}(\eta, S') = \mathrm{poly}(\mathrm{poly}(n, \log t), S + N \cdot \mathrm{poly}(m, \log N)) = \mathrm{poly}(t)$. This gives a total prover runtime of $\mathrm{poly}(t)$.
- The proof string σ has length $|\sigma| = |\mathsf{rt}| + |\sigma_{\mathcal{M}^{\otimes N}}| + |\sigma_{\mathsf{JKKZ}}| = \mathrm{poly}(\kappa) + \mathrm{poly}(\lambda, m, \log N) + D \cdot \mathrm{poly}(\eta, \log S) = D \cdot \mathrm{poly}(n, \lambda, \log t)$.
- The verifier runs $\mathcal{V}_{\mathsf{JKKZ}}$, which runs in time $(D + |\mathsf{crs}_{\mathcal{M}^{\otimes N}}| + |\langle Y \rangle| + |\sigma_{\mathcal{M}^{\otimes N}}|) \cdot \mathrm{poly}(\eta, \log S') = (D + \mathrm{poly}(\lambda, m, \log N)) \cdot \mathrm{poly}(\eta, \log(S + N \cdot \mathrm{poly}(m, \log N))) = D \cdot \mathrm{poly}(n, \lambda, \log t)$.

□

Corollary 5.11. *For* $\mathrm{poly}(n) \leq t \leq \exp(n)$ *and* $s = s(n) \geq \log t(n)$*, there exists a non-interactive argument for* $\mathsf{N}\mathcal{L}_{\mathcal{U}}(t, s)$ *that is sound against* $\mathrm{poly}(2^s)$*-size cheating provers with probability* $1 - \mathrm{negl}(2^s)$*, assuming that*

- LWE *is sub-exponentially hard,*
- *there is a* Σ*-sound SNARG for* $\mathcal{M}^{\otimes N}$*,*
- *the* $\mathcal{M}^{\otimes N}$ *verifier* $\mathcal{V}_{\mathcal{M}^{\otimes N}}$ *is a log-space uniform circuit of depth* D*.*

The honest prover runs in time $\mathrm{poly}(t)$*, the verifier runs in time* $D \cdot \mathrm{poly}(n, \lambda, s)$*, and the communication complexity is* $D \cdot \mathrm{poly}(n, \lambda, s)$*.*

Proof. The non-interactive argument for $\mathcal{L}_{\mathcal{U}}(t)$ is that given in Figure 6, with $\epsilon > 0$ corresponding to the LWE assumption and a 2^s-computational non-signaling PCP that is verified via $N = \mathrm{poly}(t)$ tests as given in Theorem 2.7. The runtime analysis is analogous to the proof of Corollary 5.10, except $\kappa = \mathrm{poly}(\log 2^s) = \mathrm{poly}(s)$, which gives that $\eta = \mathrm{poly}(n, \lambda, s)$. This gives the claimed runtimes. □

References

[Bar01a] Boaz Barak. How to go beyond the black-box simulation barrier. In *FOCS*, volume 0, pages 106–115, 2001. 1

[Bar01b] Boaz Barak. How to go beyond the black-box simulation barrier. In *FOCS*, pages 106–115, 2001. 4

[BBH+19] James Bartusek, Liron Bronfman, Justin Holmgren, Fermi Ma, and Ron D. Rothblum. On the (in)security of kilian-based snargs. In Dennis Hofheinz and Alon Rosen, editors, *Theory of Cryptography - 17th International Conference, TCC 2019, Nuremberg, Germany, December 1-5, 2019, Proceedings, Part II*, volume 11892 of *Lecture Notes in Computer Science*, pages 522–551. Springer, 2019. 1, 3, 4

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 21–31, 1991. 7

[BHK17] Zvika Brakerski, Justin Holmgren, and Yael Tauman Kalai. Non-interactive delegation and batch NP verification from standard computational assumptions. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 474–482, 2017. 3, 5, 6, 9, 10, 13

[BKK+18] Saikrishna Badrinarayanan, Yael Tauman Kalai, Dakshita Khurana, Amit Sahai, and Daniel Wichs. Succinct delegation for low-space non-deterministic computation. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 709–721, 2018. 3, 5, 7, 10

[Blu86] Manuel Blum. How to prove a theorem so no one else can claim it. In *Proceedings of the International Congress of Mathematicians*, pages 1444–1451, 1986. 1

[BMW99] Ingrid Biehl, Bernd Meyer, and Susanne Wetzel. Ensuring the integrity of agent-based computations by short proofs. In *Proceedings of the Second International Workshop on Mobile Agents*, MA '98, pages 183–194, London, UK, UK, 1999. Springer-Verlag. 3

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. In *FOCS*, pages 97–106, 2011. 13

[CCH+19] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. Fiat-shamir: from practice to theory. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 1082–1090. ACM, 2019. 1

[CCRR18] Ran Canetti, Yilei Chen, Leonid Reyzin, and Ron D. Rothblum. Fiat-shamir and correlation intractability from strong kdm-secure encryption. In *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part I*, pages 91–122, 2018. 1

[CGKS95] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, USA, 23-25 October 1995*, pages 41–50, 1995. 13

[CJJ21] Arka Rai Choudhuri, Abhishek Jain, and Zhengzhong Jin. SNARGs for P from LWE. *IACR Cryptol. ePrint Arch.*, 2021. 3, 24

[CMSZ21] Alessandro Chiesa, Fermi Ma, Nicholas Spooner, and Mark Zhandry. Post-quantum succinct arguments. *IACR Cryptol. ePrint Arch.*, 2021:334, 2021. 2, 4

[CSW20] Ran Canetti, Pratik Sarkar, and Xiao Wang. Triply adaptive UC NIZK. *IACR Cryptol. ePrint Arch.*, 2020:1212, 2020. 2

[DGI+19] Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. Trapdoor hash functions and their applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part III*, volume 11694 of *Lecture Notes in Computer Science*, pages 3–32. Springer, 2019. 13

[DHRW16] Yevgeniy Dodis, Shai Halevi, Ron D. Rothblum, and Daniel Wichs. Spooky encryption and its applications. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part III*, pages 93–122, 2016. 6

[DLN+04] Cynthia Dwork, Michael Langberg, Moni Naor, Kobbi Nissim, and Omer Reingold. Succinct proofs for NP and spooky interactions. Unpublished manuscript, 2004. http://www.cs.bgu.ac.il/~kobbi/papers/spooky_sub_crypto.pdf. 6

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986. 1

[GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the fiat-shamir paradigm. In *FOCS*, pages 102–, 2003. 4

[GK05] Shafi Goldwasser and Yael Tauman Kalai. On the impossibility of obfuscation with auxiliary input. In Éva Tardos, editor, *46th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 553–562. IEEE Computer Society, 2005. 1

[GK16] Shafi Goldwasser and Yael Tauman Kalai. Cryptographic assumptions: A position paper. In *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part I*, pages 505–522, 2016. 8

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In *STOC*, pages 113–122, 2008. 1

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity, or all languages in np have zero-knowledge proof systems. *Journal of the ACM*, 38(1):691–729, 1991. 1

[GR05] Craig Gentry and Zulfikar Ramzan. Single-database private information retrieval with constant communication rate. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings*, volume 3580 of *Lecture Notes in Computer Science*, pages 803–815. Springer, 2005. 13

[HL18] Justin Holmgren and Alex Lombardi. Cryptographic hashing from strong one-way functions (or: One-way product functions and their applications). In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 850–858. IEEE Computer Society, 2018. 1

[HLR21] Justin Holmgren, Alex Lombardi, and Ron D. Rothblum. Fiat-shamir via list-recoverable codes (or: Parallel repetition of gmw is not zero-knowledge). Cryptology ePrint Archive, Report 2021/286, 2021. https://eprint.iacr.org/2021/286. 1

[HW15] Pavel Hubácek and Daniel Wichs. On the communication complexity of secure function evaluation with long output. In Tim Roughgarden, editor, *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015, Rehovot, Israel, January 11-13, 2015*, pages 163–172. ACM, 2015. 2, 5, 11, 12

[JKKZ20] Ruta Jawale, Yael Tauman Kalai, Dakshita Khurana, and Rachel Zhang. Snargs for bounded depth computations and PPAD hardness from sub-exponential LWE. *IACR Cryptol. ePrint Arch.*, 2020:980, 2020. 7

[JKKZ21] Ruta Jawale, Yael Tauman Kalai, Dakshita Khurana, and Rachel Zhang. Snargs for bounded depth computations and PPAD hardness from sub-exponential LWE. 2021. 1, 28, 29

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *STOC*, pages 723–732, 1992. 2

[KO97] Eyal Kushilevitz and Rafail Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *FOCS*, pages 364–373, 1997. 13

[KPY19] Yael Tauman Kalai, Omer Paneth, and Lisa Yang. How to delegate computations publicly. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 1115–1124. ACM, 2019. 6

[KRR13] Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. Delegation for bounded space. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 565–574, 2013. 3, 6

[KRR14] Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. How to delegate computations: the power of no-signaling proofs. In *STOC*, pages 485–494. ACM, 2014. 3, 5, 6, 7, 9, 13, 15

[KRR17] Yael Tauman Kalai, Guy N. Rothblum, and Ron D. Rothblum. From obfuscation to the security of fiat-shamir for proofs. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 224–251. Springer, 2017. 1

[Lip05] Helger Lipmaa. An oblivious transfer protocol with log-squared communication. In Jianying Zhou, Javier López, Robert H. Deng, and Feng Bao, editors, *Information Security, 8th International Conference, ISC 2005, Singapore, September 20-23, 2005, Proceedings*, volume 3650 of *Lecture Notes in Computer Science*, pages 314–328. Springer, 2005. 13

[PS19] Chris Peikert and Sina Shiehian. Noninteractive zero knowledge for NP from (plain) learning with errors. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019, Proceedings, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 89–114. Springer, 2019. 1

[Unr12] Dominique Unruh. Quantum proofs of knowledge. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, volume 7237 of *Lecture Notes in Computer Science*, pages 135–152. Springer, 2012. 2

[Wat09] John Watrous. Zero-knowledge against quantum attacks. *SIAM J. Comput.*, 39(1):25–58, 2009. 2