Somewhere Statistical Soundness, Post-Quantum Security, and SNARGs

Yael Tauman Kalai[1], Vinod Vaikuntanathan[2], and Rachel Yun Zhang[2]

[1] Microsoft Research, Cambridge MA 02142, USA
yael@microsoft.com
[2] Massachusetts Institute of Technology, Cambridge MA 02138, USA
vinodv@csail.mit.edu, rachelyz@mit.edu

Abstract. The main conceptual contribution of this paper is a unification of two leading paradigms for constructing succinct argument systems, namely Kilian's protocol and the BMW (Biehl-Meyer-Wetzel) heuristic. We define the notion of a *multi-extractable somewhere statistically binding* (meSSB) *hash family*, an extension of the notion of somewhere statistically binding hash functions (Hubacek and Wichs, ITCS 2015), and construct it from LWE. We show that when instantiating Kilian's protocol with a meSSB hash family, the first two messages are simply an instantiation of the BMW heuristic. Therefore, if we also instantiate it with a PCP for which the BMW heuristic is sound, e.g., a computational non-signaling PCP, then the first two messages of the Kilian protocol is a sound instantiation of the BMW heuristic.

This leads us to two technical results. First, we show how to efficiently convert any succinct non-interactive argument (SNARG) for BatchNP into a SNARG for any language that has a computational non-signaling PCP. Put together with the recent and independent result of Choudhuri, Jain and Jin (Eprint 2021/808) which constructs a SNARG for BatchNP from LWE, we get a SNARG for any language that has a computational non-signaling PCP, including any language in P, but also any language in NTISP (non-deterministic bounded space), from LWE.

Second, we 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), and

- prove that Kilian's protocol, instantiated as above, is an SSS argument;
- 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; and
- conjecture that constant-round SSS arguments can be soundly converted into non-interactive arguments via the Fiat-Shamir transformation.

Keywords: SNARGs · Fiat-Shamir · Kilian · Post-quantum security · Straight-line soundness.

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).[3] In our work, we propose two paths towards obtaining SNARGs for P as well as for certain sub-classes of NP. Our approaches are motivated by Kilian's celebrated work [Kil92] that converts any PCP into an interactive argument using a tree hash [Mer87].

Recall that in Kilian's protocol, the prover tree-commits to a PCP using a hash key generated by the verifier and sends the resulting commitment to the verifier. The verifier then samples a PCP query at random and sends it to the prover. The prover must then send back answers to the PCP queries along with verification paths for each answer w.r.t. the previously sent commitment.

We take a somewhat anachronistic view and see Kilian's *four-message, public-coin* interactive argument as a natural interpolation of the *two-message, privately verifiable* Biehl-Meyer-Wetzel (BMW) heuristic.

Recall that the BMW heuristic takes any PCP and any (computationally secure) single-sever PIR scheme, and uses them to construct a two-message succinct argument where the verifier sends each PCP query to the prover as a PIR query, and the prover runs the PIR protocol with the database being the PCP proof string, and responds accordingly (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 it is instantiated with a PCP with a special property known as computational non-signaling [KRR13,BHK17]. We note that not all NP languages have such a (computational non-signaling) PCP, and such a PCP was constructed only for P [KRR14,BHK17] and some sub-classes of NP such as NTISP [BKK+18].[4] We refer to such PCPs for which the BMW heuristic is computationally sound as BMW-*compatible*. Note that the BMW heuristic results in a privately verifiable protocol since the verifier needs to run the PIR decoding algorithm on the prover's message, in a sense decrypting it using a private key.

Constructing two-message *publicly verifiable* succinct arguments, which in turn give us SNARGs in the common reference string model, appears to be a significantly harder challenge. Indeed, the only construction we have of SNARGs under a post-quantum assumption is restricted to bounded depth computations [JKKZ20]. One attempt to constructing a SNARG for all of P was recently made in [KPY19], which showed how to convert the BMW heuristic to a publicly verifiable one by relying on a primitive called *zero-testable encryption* [PR17]. In addition, they gave a construction of this primitive 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, a problem which we tackle in this work.

1.1 Multi-Extractable Somewhere Statistically Binding (meSSB) Hash Families

As a starting point, consider instantiating Kilian's protocol with a somewhere statistically binding (SSB) hash function [HW15] in place of a generic tree hash. Recall that an SSB hash family is a hash family $\mathcal{H}$ where each hash key hk is associated with an index $i \in [L]$, where L is the length of the input, such that $\mathsf{Hash}(\mathsf{hk}, x)$ is statistically binding on x_i, and importantly, the key hk hides the index i. In this work we consider *extractable* SSB (eSSB) hash families, which are SSB hash families with the additional property that one can extract x_i from the hash value $\mathsf{Hash}(\mathsf{hk}, x)$ given a trapdoor td that is generated together with hk (see Section 2.3, Definition 9).

We observe that an eSSB hash family is essentially a (computational) single-server PIR scheme, where the query corresponding to index i is the hash key hk associated with the index i, the database answer corresponding to database x simply runs $\mathsf{Hash}(\mathsf{hk}, x)$, and given the trapdoor td corresponding to hk one

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

[4] 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. The computational non-signaling PCP constructed in [BKK+18] has query complexity $s \cdot \mathrm{polylog}(t)$ and as a result the communication complexity of the BMW heuristic grows with $s \cdot \mathrm{polylog}(t)$.

can indeed extract x_i from $\mathsf{Hash}(\mathsf{hk}, x)$, without revealing the secret index i. Armed with this observation, we note that if we instantiate Kilian's protocol with an eSSB hash family, the first two messages are quite similar to the BMW heuristic, the difference being that the BMW heuristic uses many PIR queries (as many as the number of PCP queries), whereas an eSSB families support a single PIR query.

To remedy this, we consider the notion of a *multi-extractable* SSB (meSSB) hash family, where each key hk is associated with several indices $i_1, \ldots, i_\ell \in [N]$, and is generated with trapdoors $\mathsf{td}_1, \ldots, \mathsf{td}_\ell$, such that one can extract $x_{i_1}, \ldots, x_{i_\ell}$ from $\mathsf{Hash}(\mathsf{hk}, x)$. Importantly, since in the BMW heuristic each query is generated using fresh randomness, to match this heuristic, we need to require that for every $i \in [\ell]$, the index i remains hidden, even given the key hk and all the trapdoors $\{\mathsf{td}_j\}_{j \in [\ell] \setminus \{i\}}$. We note that if we instantiate Kilian's protocol with a meSSB hash family then the first two messages are precisely the BMW heuristic! This observation is the first conceptual contribution of this work.

This instantiation of Kilian's protocol with a meSSB hash family can alternatively be thought of as a way of converting the BMW protocol to a publicly verifiable one, albeit at the cost of adding two rounds. In this instantiation, we execute the BMW heuristic, but *the verifier never decrypts the* PIR *answers*. Instead, we view the PIR answers 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.

Starting with this observation, we proceed to offer two paths to convert Kilian's protocol into a SNARG.

The Fiat-Shamir Paradigm. The first approach, which we elaborate on in Section 1.2, considers applying the Fiat-Shamir paradigm to Kilian's protocol when instantiated with a meSSB hash family and with a BMW-compatible PCP.[5] At first it may seem that this approach is doomed to fail, in light of the recent negative result of [BBH+19], which shows that the Fiat-Shamir paradigm is not sound when applied to Kilian's protocol. However, we argue that this specific instantiation of Kilian's protocol has a special property, which we refer to as *somewhere statistical soundness* (SSS), that allows it to evade this specific negative result. We conjecture that SSS protocols are Fiat-Shamir friendly, meaning that for any SSS protocol there is some choice of hash function using which the application of the Fiat-Shamir paradigm to the protocol is sound. Additionally, we argue that SSS protocols are of independent interest. In particular, we prove that every SSS protocol has a straight-line soundness proof and as a result is post-quantum sound (assuming the underlying assumption is post-quantum secure). We elaborate on this in Section 1.2.

Using SNARGs for BatchNP. The second approach we consider is to use a SNARG for BatchNP to convert the first two messages in the above instantiation of Kilian's protocol for a language L into a SNARG for L. A similar result was shown independently by [CJJ21]. We elaborate on this in Section 1.3.

1.2 Somewhere Statistically Sound (SSS) Interactive Arguments

One noteworthy property of our instantiation of Kilian's protocol is that due to the soundness of the BMW heuristic, with high probability a cheating prover is statistically committed to incorrect answers on the particular locations specified by the meSSB hash function. Thus, if the verifier's PCP query points to exactly the locations that are statistically bound by the meSSB hash function, the verifier is guaranteed to reject no matter what the final message of the prover is. We call this *somewhere statistical soundness* (SSS), in analogy with somewhere statistical binding.

We can extend this to multi-round protocols just as well, however we focus on 4-round protocols where, without loss of generality, we assume that the first message is sent by the verifier. Formally, 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:

[5] We note that such a PCP may not always exist, but is known to exist for P and some sub-classes of NP.

– For every poly-size *deterministic* 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 (deterministic) poly-size $\mathcal{P}^*$, with overwhelming probability, any (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.

We study the implications of SSS protocols. 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 1.2.
2. Second, we prove that Kilian's protocol, instantiated with a meSSB hash family (for which constructions based on the LWE assumption exist) and a BMW-compatible PCP, 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."[6]
   We note that we have constructions of BMW-compatible PCPs only for deterministic languages and for specific classes of non-deterministics languages such as NTISP, and thus our instantiation of Kilian's protocol is post-quantum sound for only such classes. 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.[7]
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. We elaborate on this (and define the Fiat-Shamir paradigm) in Section 1.2. We mention that 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 indeed an SSS argument and was used to construct a (non-succinct) UC NIZK for NP with an adaptive soundness guarantees.
   We emphasize that we do not prove that any SSS interactive argument is Fiat-Shamir friendly, only conjecture it. We believe that it is a promising path for obtaining SNARGs based on a standard post-quantum assumption. In particular, we propose constructing an SSS interactive argument for all of NP as a great open problem. We note that it is easier than constructing a (non-adaptive) SNARG for NP, since any such SNARG is in particular SSS (with two additional arbitrary rounds that are ignored by the verdict function). Constructing a non-adaptive SNARG for NP has been a major open problem, and constructing a succinct SSS protocol can be seen as a stepping stone for achieving this goal.
   Note that since we prove that our instantiation of Kilian's protocol is SSS, and since we conjecture that any SSS protocol is Fiat-Shamir friendly, as a special case we conjecture that our instantiation of Kilian's protocol (with a meSSB hash family and a BMW-compatible PCP) is Fiat-Shamir friendly.

[6] https://simons.berkeley.edu/events/quantum-lectures-crypto-dummies
[7] We mention that it is not clear how to simplify their proof for the subclasses of NP as above, without using a meSSB hash family (rather using an arbitrary collapse binding hash family), since our simple post-quantum proof strongly relies on the fact that the first two messages are a sound instantiation of the BMW heuristic.

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 to use a meSSB hash family *combined with* a BMW-compatible PCP. If sound, this would yield a SNARG for all of P (and some sub-classes of NP, as described above).

SSS, Straight-Line Soundness and Post-Quantum Security 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 (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 straight-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 receiving $\alpha_1 = \mathcal{P}^*(\beta_1)$, 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 (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.

A formal proof of this Claim appears in Section 4.2 (Theorem 9). This property makes SSS arguments particularly appealing, given the major effort by the community to make cryptographic protocols post-quantum secure.

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." Recall that 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 [Bar01,GK05]. Moreover, recently it was shown that the Fiat-Shamir paradigm is not sound when applied to the celebrated Kilian's protocol [BBH+19].

As a natural interpolation between statistically sound proofs and computationally sound arguments, it is natural to ask whether the hybrid class of all (constant round) SSS interactive arguments is Fiat-Shamir friendly.

Conjecture 1. Any constant round SSS interactive argument $(\mathcal{P}, \mathcal{V})$ is *Fiat-Shamir friendly*.

We note that all known negative results for the Fiat-Shamir paradigm [Bar01,GK03,BBH+19] 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 eSSB 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 meSSB is sufficient to apply Fiat-Shamir soundly, assuming the PCP in use makes the BMW heuristic sound.

Instantiating an SSS version of Kilian We show that Kilian's protocol instantiated with a meSSB hash family, and a BMW-compatible PCP, is an SSS argument. In particular, we obtain the following corollary.

Theorem 2 (Informal). *Kilian's protocol is* SSS*, and thus has post-quantum soundness, if we use a* BMW*-compatible* PCP *and if the prover commits to this* PCP *using a post-quantum* meSSB *hash function.*

Hubáček and Wichs [HW15] constructed an eSSB hash family assuming the hardness of LWE. This hash family is post-quantum secure assuming the post-quantum hardness of LWE. We note that any eSSB hash family can be easily extended to a meSSB hash family.

Moreover, (adaptive) BMW-compatible PCPs are known for all deterministic languages [BHK17] and languages in NTISP [BKK+18]. For deterministic languages the (adaptive) soundness relies on the polynomial hardness of the underlying PIR scheme, whereas for languages in NTISP the (adaptive) soundness relies on the sub-exponential hardness of the underlying PIR scheme.[8] The query complexity for languages in DTIME(t)

[8] We mention that the difference stems from the fact that for the non-deterministic languages we can only construct PCPs that have sub-exponential non-signaling (adaptive) soundness, whereas deterministic languages have polynomial non-signaling (adaptive) soundness.

is $\mathrm{polylog}(t)$, and for languages in $\mathsf{NTISP}(t,s)$ it is $s \cdot \mathrm{polylog}(t)$. These results, together with Theorem 2, imply the following corollary.

Corollary 1 (Informal). *There exists an instantiation of Kilian's protocol that is* SSS*, and thus post-quantum sound, for all deterministic computations assuming the polynomial post-quantum hardness of* LWE*, and for all languages in* NTISP *assuming the sub-exponential post-quantum hardness of* LWE*. For* $\mathsf{DTIME}(t)$ *languages the communication complexity grows with* $\mathrm{polylog}(t)$*, and for languages in* $\mathsf{NTISP}(t,s)$ *the communication complexity grows with* $s \cdot \mathrm{polylog}(t)$*.*

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 SNARGs: from BatchNP to P and Beyond

This view of the first two messages of Kilian's protocol as an instantiation of the BMW heuristic leads us to our final contribution: an alternative pathway to getting a SNARG for any language that has a BMW-compatible PCP. Specifically, we show a reduction from constructing a SNARG for the class of all languages that have a BMW-compatible PCP to the simpler goal of constructing a SNARG for BatchNP.

The starting point is the two-round preamble where the verifier sends the prover the description of a meSSB hash function, and the prover replies with a multi-extractable commitment to a BMW-compatible 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 a valid opening; 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 the BMW-heuristic the answers that are committed to by the meSSB hash function are rejecting, and hence by the meSSB 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 super-polynomially large, meaning that the (honest) prover runtime is super-polynomial. Fortunately, all known PCP constructions (including the ones from [KRR14,BHK17,BK18]) 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.[9] 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 opening; 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.

[9] 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.

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 hash value, *after* he receives the CRS/first message of the BatchNP proof. Since the hash value is 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 6.

Concurrent Work. In a concurrent and independent work, Choudhuri, Jain and Jin [CJJ21] construct SNARGs for BatchNP from LWE. Thus, using their result together with our reduction from Section 1.3, we obtain a SNARG for any language that has a BMW-compatible PCP, from the LWE assumption. In particular, as we elaborate on in Section 6, we obtain a SNARG for any language in $\mathsf{DTIME}(t)$ or in $\mathsf{NTISP}(t, s)$ 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.

2 Preliminaries

Definition 1. *Two distribution ensembles $\{A_k\}_{k\in\mathbb{N}}$ and $\{B_k\}_{k\in\mathbb{N}}$ are said to be Ω-indistinguishable if for every $\mathrm{poly}(\Omega)$-size distinguisher $\mathcal{D}$ there exists a negligible function μ such that for every $k \in \mathbb{N}$,*

$$|\Pr_{a\leftarrow A_k}[\mathcal{D}(a) = 1] - \Pr_{b\leftarrow B_k}[\mathcal{D}(b) = 1]| \leq \mu(\Omega(k)).$$

2.1 Straight-Line Reductions

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

Definition 2. *(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 (adaptively) $\theta = \theta(\kappa)$-*straight-line sound *if there is a* PPT *black box reduction $\mathcal{R}$ and a non-interactive θ-decisional complexity assumption* [GK16],[10] *such that $\mathcal{R}$, given oracle access to any cheating prover $\mathcal{P}^*$ that breaks (adaptive) 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*[11] *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 3 ([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 $\kappa \in \mathbb{N}$,*

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

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

2.2 Probabilistically Checkable Proofs (PCP)

We first recall the definition of a *probabilistically checkable proof* (PCP). A PCP for an NP language $\mathcal{L}$ is a (deterministic) function Π that takes as input a witness w for a statement $x \in L$, and converts it into a proof $\pi = \Pi(x, w)$ which can be verified by a randomized verifier that reads only a few of its bits.

Definition 4 (PCP). *A probabilistically 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 will denote the length of the PCP by $L = |\pi|$.*
- *$\mathcal{Q}_{\mathsf{PCP}}$ 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 5. *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 1. We also consider a stronger notion of PCP soundness known as *non-signaling soundness*, and more specifically *computational non-signaling soundness*. The precise definition (given in Appendix A) is not needed in order to understand our result: what is important is that computational non-signaling PCPs are BMW-compatible.

Two remarks are in place. First, two flavors of (computational) non-signaling soundness have been considered in the literature: adaptive and non-adaptive; the latter provides non-adaptive soundness of the BMW heuristic, whereas the former provides adaptive soundness. In this work, we will describe the results with adaptive soundness. Second, 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 Appendix A for the precise definitions.

Adaptive 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 3 ([KRR14,BHK17]). *For any $\mathrm{poly}(n) \leq t(n) \leq \exp(n)$, there exists an adaptive 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 5).*

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

Theorem 4 ([BKK+18]). *For $\mathrm{poly}(n) \leq t \leq \exp(n)$ and $s = s(n) \geq \log t(n)$, there is an adaptive 2^s-computational non-signaling* PCP *for $\mathsf{N}\mathcal{L}_{\mathcal{U}}(t,s)$ 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.*

2.3 Hash Function Families with Local Opening

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

Definition 6 (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 7 (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 $\mathsf{hk} \leftarrow \mathsf{Gen}(1^\kappa, L)$.

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 $i \in [\ell]$, $\mathsf{Verify}(\mathsf{hk}, \mathsf{rt}, q_i, \pi_{q_i}, \mathsf{o}_i) = 1$.

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

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 8 ([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 5 ([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).*

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.

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 $V_{\mathsf{PCP}}(x, (q_1, \ldots, q_\ell), (a_1, \ldots, a_\ell)) = 1$.

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

3 Somewhere Statistically Binding Hash Functions

Central to our paper is the notion of somewhere statistically binding (SSB) hash functions, first defined by Hubáček and Wichs [HW15]. These are hash functions with local openings that have an additional special property: for any index i^*, one can generate a hash key that guarantees statistical binding for the position i^*. Namely, even an unbounded adversary cannot open the bit at position i^* to two different values. Furthermore, the hash key should be index-hiding, namely, it should hide the index i^* from all polynomial-time adversaries.

We augment this notion in two ways. First, we require that the statistically bound value at position i^* can be recovered from the hash output using a trapdoor underlying the hash key. It turns out that the Hubáček-Wichs construction of SSB hash functions from homomorphic encryption already satisfies this property. Secondly, we augment the SSB family so that the hash key guarantees statistical binding for a set of positions I simultaneously. Extractability now requires that there are $|I|$ trapdoors, where td_i helps us recover the statistically bound value at the i^{th} position for any $i \in I$. Finally, the index-hiding property needs

to be augmented to hold even given the other trapdoors. We call the resulting notion a multi-extractable SSB (or meSSB) hash function.

We first present the definition of extractable SSB (eSSB) hash functions in Section 3.1 and that of multi-extractable SSB (meSSB) hash functions in Section 3.2. We also show how to construct meSSB hash functions from any eSSB hash function family in a simple way. Finally, in Section 3.3 and Appendix A, we reprove the soundness of the BMW protocol when instantiated with a meSSB hash function.

3.1 Extractable Somewhere Statistically Binding (eSSB) Hash Functions

Definition 9 (eSSB Hash Family). *An $S = S(\kappa)$-hiding* extractable somewhere statistically binding (eSSB) hash family *is a hash family with local opening* $(\mathsf{Gen}, \mathsf{Hash}, \mathsf{Open}, \mathsf{Verify})$*, with the following changes:*

- $\mathsf{Gen}(1^\kappa, L, i)$ *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 $\mathsf{td} \in \{0,1\}^{\ell_{\mathsf{td}}}$,*

An eSSB *hash family also has 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 $\mathsf{poly}(S(\kappa))$-size adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ there exists a negligible function μ such that for any $L \leq 2^\kappa$,*

$$\Pr\left[b = b' \;\middle|\; \begin{array}{r} i_0, i_1, \mathsf{state} \leftarrow \mathcal{A}_1(1^\kappa) \\ b \xleftarrow{\$} \{0,1\} \\ (\mathsf{hk}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\kappa, L, i_b) \\ b' \leftarrow \mathcal{A}_2(\mathsf{hk}, \mathsf{state}) \end{array}\right] = \frac{1}{2} + \mu(S(\kappa)).$$

- **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 $(\mathsf{hk}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\kappa, L, i)$.
- **Somewhere Statistically Binding:** *For any $\kappa \in \mathbb{N}$, $L \leq 2^\kappa$, $i \in [L]$ and $\mathsf{rt} \in \{0,1\}^{\ell_{\mathsf{hash}}}$,*

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

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

Remark 2. We note 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 $\mathsf{rt} \in \{0,1\}^{\ell_{\mathsf{hash}}}$,

$$\Pr[\exists \; (u, \mathsf{o}, \mathsf{o}') \;\; \text{s.t.} \;\; u \neq u' \;\; \wedge \;\; \mathsf{Verify}(\mathsf{hk}, \mathsf{rt}, i, u, \mathsf{o}) = \mathsf{Verify}(\mathsf{hk}, \mathsf{rt}, i, u', \mathsf{o}') = 1] = 0,$$

where the probability is over $\mathsf{hk} \leftarrow \mathsf{Gen}(1^\kappa, L, i)$. The difference is that our definition permits "invalid" hash values for which Invert outputs $\perp$, and we require that such hash values have no valid openings. The [HW15] definition simply requires that there is at most one valid opening for every hash value. This distinction, however, is not crucial to the rest of our paper.

Hubáček and Wichs constructed SSB hash functions assuming the existence of a leveled homomorphic encryption scheme, and their construction is an extractable SSB hash function as well. We state the formal theorem below.

Theorem 6 ([HW15]). *Assuming the sub-exponential hardness of the learning with errors (*LWE*) problem, there exists a 2^{κ^ϵ}-hiding* eSSB *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).*

3.2 Multi-Extractable SSB (meSSB) Hash Functions

Definition 10. *An $S = S(\kappa)$-hiding multi-extractable somewhere statistically binding (*meSSB*) hash family is a hash family with local opening* $(\mathsf{Gen}, \mathsf{Hash}, \mathsf{Open}, \mathsf{Verify})$*, where*

- $\mathsf{Gen}(1^\kappa, L, \ell, I)$ *takes as additional input ℓ locations $I = (i_1, \ldots, i_\ell) \in [L]^\ell$ and outputs a hash key $\mathsf{hk} \in \{0,1\}^{\ell_{\mathsf{hk}}}$ as well as a trapdoor $\mathsf{td} = (\mathsf{td}_1, \ldots, \mathsf{td}_\ell) \in \{0,1\}^{\ell \cdot \ell_{\mathsf{td}}}$,*

along with an additional PPT *algorithm* Invert *which works as follows.*

- $\mathsf{Invert}(J, \{\mathsf{td}_j\}_{j\in J}, \mathsf{rt})$ *takes as input a subset $J \subseteq [\ell]$ of indices as well as a partial trapdoor $\{\mathsf{td}_j\}_{j\in J}$ and a hash value $\mathsf{rt} \in \{0,1\}^{\ell_{\mathsf{hash}}}$, and outputs $u \in \{0,1,\perp\}^{|J|}$.*
  When no subset J is provided, $\mathsf{Invert}(\mathsf{td}, \mathsf{rt})$ *takes as input a full trapdoor $\mathsf{td} \in \{0,1\}^{\ell \cdot \ell_{\mathsf{td}}}$ and a hash value $\mathsf{rt} \in \{0,1\}^{\ell_{\mathsf{hash}}}$ and outputs $u \in \{0,1,\perp\}^{\ell}$.*

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 $L \le 2^\kappa$,*

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

  In words, index-hiding requires that even given the trapdoor information for the overlap of the two ordered sets $I^0 = (i^0_1, \ldots, i^0_\ell)$ and $I^1 = (i^1_1, \ldots, i^1_\ell)$, the adversary still cannot distinguish whether hk *is statistically binding on I^0 or I^1.*
- **Correctness of Inversion:** *For any $\kappa \in \mathbb{N}$, $L \le 2^\kappa$, and any $I \in [L]^\ell$, $J \subseteq [\ell]$, and $x \in \{0,1\}^L$,*

$$\Pr[\mathsf{Invert}(J, \{\mathsf{td}_j\}_{j\in J}, \mathsf{Hash}(\mathsf{hk}, x)) = \{x_{i_j}\}_{j\in J}] = 1,$$

  where the probability is over $(\mathsf{hk}, \mathsf{td}) \leftarrow \mathsf{Gen}(1^\kappa, L, \ell, I)$.
- **Somewhere Statistically Binding:** *For any $\kappa \in \mathbb{N}$, $L \le 2^\kappa$, $I \in [L]^\ell$, $i \in I$ and $\mathsf{rt} \in \{0,1\}^{\ell_{\mathsf{hash}}}$,*

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

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

Multi-extractable SSB (meSSB) hash families can be constructed from extractable SSB (eSSB) families by invoking many independent copies. The formal construction is given in Figure 3.

An meSSB Hash Family

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

$$\mathcal{H}_{\mathsf{meSSB}} = (\mathsf{Gen}_{\mathsf{meSSB}}, \mathsf{Hash}_{\mathsf{meSSB}}, \mathsf{Open}_{\mathsf{meSSB}}, \mathsf{Verify}_{\mathsf{meSSB}}, \mathsf{Invert}_{\mathsf{meSSB}}),$$

is defined as follows:

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

Fig. 3. The meSSB Hash Family $(\mathsf{Gen}_{\mathsf{meSSB}}, \mathsf{Hash}_{\mathsf{meSSB}}, \mathsf{Open}_{\mathsf{meSSB}}, \mathsf{Verify}_{\mathsf{meSSB}}, \mathsf{Invert}_{\mathsf{meSSB}})$

Lemma 1. *When the number of statistically bound locations ℓ is at most $\mathrm{poly}(S)$, the hash family $\mathcal{H}_{\mathsf{meSSB}}$ defined in Figure 3 is an S-hiding multi-extractable* SSB *hash family. Furthermore, its S-hiding is S-straight line reducible from the S-hiding of the underlying* eSSB *hash family. It also holds that*

$$\ell_{\mathsf{meSSB,hk}} = \ell \cdot \ell_{\mathsf{eSSB,hk}}, \ \ell_{\mathsf{meSSB,hash}} = \ell \cdot \ell_{\mathsf{meSSB,hash}}, \ \ell_{\mathsf{meSSB,td}} = \ell \cdot \ell_{\mathsf{eSSB,td}}, \ \textit{and} \ \ell_{\mathsf{meSSB,o}} = \ell \cdot \ell_{\mathsf{meSSB,o}}$$

where $\ell_{\mathsf{meSSB,hk}}, \ell_{\mathsf{meSSB,hash}}, \ell_{\mathsf{meSSB,td}}, \ell_{\mathsf{meSSB,o}}$ are the parameters associated with $\mathcal{H}_{\mathsf{meSSB}}$ and $\ell_{\mathsf{eSSB,hk}}, \ell_{\mathsf{eSSB,hash}}, \ell_{\mathsf{eSSB,td}}, \ell_{\mathsf{eSSB,o}}$ are the parameters associated with $\mathcal{H}_{\mathsf{eSSB}}$.

Proof. The correctness of inversion and ℓ-somewhere statistically binding properties follow straightforwardly from the corresponding properties of the underlying eSSB hash family (Definition 9), so we focus on the S-index hiding property. In particular, we present a straight-line reduction from the S-hiding of the underlying eSSB hash family to the S-hiding of the meSSB 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 $\mathsf{hk}_{\mathsf{meSSB}}$ 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, given partial trapdoor information $\mathsf{td}_{\mathsf{meSSB}}|_{i^0 \cap i^1}$. 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 $j^* \notin i^0 \cap i^1$ for which $\mathcal{A}_2(\cdot, \mathsf{state})$ can distinguish between $\mathsf{hk}_{\mathsf{meSSB}}$ 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 $\geq \delta(S)/\ell$. Then, to break the S-hiding of the eSSB hash family, an adversary can distinguish between $\mathsf{hk}^*_{\mathsf{eSSB}}$ generated

by $\mathsf{Gen}_{\mathsf{eSSB}}(1^\kappa, L, i^0_{j^*})$ and $\mathsf{Gen}_{\mathsf{eSSB}}(1^\kappa, L, i^1_{j^*})$ by generating $(\mathsf{hk}_{\mathsf{eSSB},j}, \mathsf{td}_{\mathsf{eSSB},j}) \leftarrow \mathsf{Gen}_{\mathsf{eSSB}}(1^\kappa, L, i^{b(j)}_j)$ 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 meSSB hash key $(\mathsf{hk}_{\mathsf{eSSB},1}, \ldots, \mathsf{hk}_{\mathsf{eSSB},j^*-1}, \mathsf{hk}^*_{\mathsf{eSSB}}, \mathsf{hk}_{\mathsf{eSSB},j^*+1}, \ldots, \mathsf{hk}_{\mathsf{eSSB},\ell})$ and outputs $\mathcal{A}_2(\cdot, \mathsf{state})$'s output. This has a distinguishing advantage of $\geq \delta(S)/\ell$, which is non-negligible in S.

Finally, observe that this reduction is straight-line.

3.3 The BMW Protocol with meSSB Hash Families

Recall that the BMW heuristic is a two message succinct argument, where the verifier queries a PCP via a PCP query consisting of ℓ locations by sending ℓ parallel independent PIR queries to the prover. The prover computes, under the PIR, the ℓ answers and sends them back to the verifier. The verifier then reconstructs the ℓ answers and checks them via the PCP verification algorithm.

We note that a eSSB hash family functions as a PIR scheme, as follows:

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

Thus, we can run the BMW heuristic with eSSB hash functions in place of the PIRs. In fact, the notion of these ℓ parallel eSSB hash functions is captured by our notion of a meSSB hash function, and thus we can run the BMW heuristic with a single meSSB hash function (binding on the ℓ locations of a PCP query) instead of the ℓ parallel PIR queries. Indeed, as we formally state below, the BMW heuristic is sound when instantiated with a meSSB hash family and a computationally non-signaling PCP.

Let $(\mathsf{Gen}_{\mathsf{meSSB}}, \mathsf{Hash}_{\mathsf{meSSB}}, \mathsf{Open}_{\mathsf{meSSB}}, \mathsf{Verify}_{\mathsf{meSSB}}, \mathsf{Open}_{\mathsf{meSSB}})$ be a meSSB hash family. 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 $(q_1, \ldots, q_\ell) \leftarrow \mathcal{Q}_{\mathsf{PCP}}(1^\kappa)$. He computes $(\mathsf{hk}_{\mathsf{meSSB}}, \mathsf{td}_{\mathsf{meSSB}}) \leftarrow \mathsf{Gen}_{\mathsf{meSSB}}(1^\kappa, L, (q_1, \ldots, q_\ell))$ and sends $\mathsf{hk}_{\mathsf{meSSB}}$ to the prover.
- **Prover:** $\mathcal{P}_{\mathsf{BMW}}$ computes the PCP string $\pi = \Pi(x)$, and sends x and $\mathsf{rt} \leftarrow \mathsf{Hash}_{\mathsf{meSSB}}(\mathsf{hk}_{\mathsf{meSSB}}, \pi)$ to the verifier.
- **Verdict:** $\mathcal{V}_{\mathsf{BMW}}$ computes $(a_1, \ldots, a_\ell) \leftarrow \mathsf{Invert}_{\mathsf{meSSB}}([\ell], \mathsf{td}_{\mathsf{meSSB}}, \mathsf{rt})$ and accepts if and only if $\mathcal{V}_{\mathsf{PCP}}(x, (q_1, \ldots, q_\ell), (a_1, \ldots, a_\ell)) = 1$.

Fig. 4. BMW Heuristic with a meSSB Hash Function

Theorem 7. *Let $(\Pi, \mathcal{Q}_{\mathsf{nsPCP}}, \mathcal{V}_{\mathsf{nsPCP}})$ be a* PCP *for a language $\mathcal{L}$ with adaptive $\Omega(n)$-computational non-signaling soundness and locality ℓ. Assume that the* meSSB *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 cheating prover $\mathcal{P}^*$, there is a negligible function μ such that*

$$\Pr\left[\mathcal{V}_{\mathsf{BMW}}(x, \mathsf{rt}, \mathsf{td}_{\mathsf{meSSB}}, (q_1, \ldots, q_\ell)) = 1 \ \wedge \ x \notin \mathcal{L}\right] \leq \mu(\Omega'),$$

where $(x, \mathsf{rt}) \leftarrow \mathcal{P}^*(\mathsf{hk}_{\mathsf{meSSB}})$ *and where the probability is over* $(q_1, \ldots, q_\ell) \leftarrow Q_{\mathsf{PCP}}(1^\kappa)$ *and* $(\mathsf{hk}_{\mathsf{meSSB}}, \mathsf{td}_{\mathsf{meSSB}}) \leftarrow \mathsf{Gen}_{\mathsf{meSSB}}(1^k, L, (q_1, \ldots, q_\ell))$*. In other words,*

$$\Pr\left[\mathcal{V}_{\mathsf{nsPCP}}(x, Q, \mathsf{Invert}_{\mathsf{meSSB}}(\mathsf{td}_{\mathsf{meSSB}}, \mathsf{rt})) = 1 \ \wedge \ x \notin \mathcal{L} \,\middle|\, \begin{array}{r} Q \leftarrow \mathcal{Q}_{\mathsf{nsPCP}}(1^\kappa, L) \\ (\mathsf{hk}_{\mathsf{meSSB}}, \mathsf{td}_{\mathsf{meSSB}}) \leftarrow \mathsf{Gen}_{\mathsf{meSSB}}(1^\kappa, L, Q) \\ (x, \mathsf{rt}) \leftarrow \mathcal{P}^*(\mathsf{hk}_{\mathsf{meSSB}}) \end{array}\right] = \mathrm{negl}(\Omega'). \quad (1)$$

Moreover, this is proven via a Ω'*-straight-line reduction (Definition 2).*

For the sake of completeness, we prove Theorem 7 in Appendix A.

4 Somewhere Statistically Sound Interactive Arguments

4.1 Defining SSS Arguments

Let κ denote a security parameter.

Definition 11. *An interactive argument* $(\mathcal{P}, \mathcal{V})(1^\kappa)$ *for a language* $\mathcal{L} = \{\mathcal{L}_n\}_{n\in\mathbb{N}}$ *is* statistically sound *if for every (potentially computationally unbounded) cheating prover* $\mathcal{P}^*$ *there exists a negligible function* μ *such that for every* $x \notin \mathcal{L}$*, the soundness error is negligible. That is,*

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

We will sometimes parameterize the soundness error and will call a protocol θ-statistically sound *if its soundness error is at most* $\theta(\kappa)$*.*

Definition 12. *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:*

- **(Adaptive)** θ**-Somewhere Statistically Soundness:** *For every* $\mathrm{poly}(\theta)$*-size (cheating) prover* $\mathcal{P}^*$ *that generates an instance* x*, 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* $\mathcal{P}^*$ *that gives* x *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 and instance 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 instance and 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.

4.2 SSS implies Straight-Line Soundness

Theorem 8. *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 adaptie 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 adaptive θ-somewhere statistically sound and θ-computational indistinguishability properties, we obtain that there is a θ-straight-line reduction from A to the adaptive θ-soundness of $(\mathcal{P}, \mathcal{V})$.

Suppose that there is a $\mathrm{poly}(\theta)$-size cheating prover $\mathcal{P}^*$ such that $\Pr[(\mathcal{P}^*, \mathcal{V})(1^\kappa, x) = 1 : x \leftarrow \mathcal{P}^*(1^\kappa)] = \delta(\theta)$, 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 for instance x produced by $\mathcal{P}^*$ is accepting, $\mathcal{R}$ outputs 1. Otherwise, it outputs 0.

Note that

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

so the distinguishing advantage of the reduction is

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

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.

4.3 SSS implies Post-Quantum Soundness

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 9. *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 $\kappa \in \mathbb{N}$, produces a rejecting instance and convinces $\mathcal{V}$ of this 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 and instance 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 $\mathcal{P}^*$.

5 Kilian's Protocol is Somewhere Statistically Sound

We instantiate Kilian's protocol with two ingredients: an adaptive $\Omega(n)$-computational non-signaling PCP $(\Pi, \mathcal{Q}_{\mathsf{nsPCP}}, \mathcal{V}_{\mathsf{nsPCP}})$ for a language $\mathcal{L}$, and a meSSB hash family $(\mathsf{Gen}_{\mathsf{meSSB}}, \mathsf{Hash}_{\mathsf{meSSB}}, \mathsf{Open}_{\mathsf{meSSB}}, \mathsf{Verify}_{\mathsf{meSSB}}, \mathsf{Invert}_{\mathsf{meSSB}})$. The resulting protocol is described in Figure 5.

Kilian's protocol with a non-signaling PCP and an meSSB 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 $(\mathsf{hk}_{\mathsf{meSSB}}, \mathsf{td}_{\mathsf{meSSB}}) \leftarrow \mathsf{Gen}_{\mathsf{meSSB}}(1^\kappa, L, Q)$, and sends $\mathsf{hk}_{\mathsf{meSSB}}$ to the prover.
- **First prover's meessage:** $\mathcal{P}_{\mathsf{nsKilian}}$ computes the PCP proof $\pi = \Pi(x)$ and its hash value $\mathsf{rt} = \mathsf{Hash}_{\mathsf{meSSB}}(\mathsf{hk}_{\mathsf{meSSB}}, \pi)$. 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 $\mathsf{o}_i = \mathsf{Open}_{\mathsf{meSSB}}(\mathsf{hk}_{\mathsf{meSSB}}, \pi, q_i)$, 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 $i \in [\ell]$, $\mathsf{Verify}_{\mathsf{meSSB}}(\mathsf{hk}_{\mathsf{meSSB}}, \mathsf{rt}, q_i, \pi_{q_i}, \mathsf{o}_i) = 1$.

Fig. 5. The Protocol $(\mathcal{P}_{\mathsf{nsKilian}}, \mathcal{V}_{\mathsf{nsKilian}})$ for $\mathcal{L}$

With these ingredients, and setting κ to be $(\log \Omega(n))^{1/\epsilon}$ such that $2^{\kappa^\epsilon} = \Omega$, the resulting Kilian's protocol is a 2^{κ^ϵ}-SSS argument assuming the meSSB hash family is 2^{κ^ϵ}-hiding, 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 2. *$(\mathcal{P}_{\mathsf{nsKilian}}, \mathcal{V}_{\mathsf{nsKilian}})$ is a Ω-SSS interactive argument assuming the* meSSB *hash family is 2^{κ^ϵ}-hiding.*

Proof. For $(\mathsf{hk}_{\mathsf{meSSB}}, \mathsf{td}_{\mathsf{meSSB}}) \leftarrow \mathsf{Gen}_{\mathsf{meSSB}}(1^\kappa, L, Q)$, define $T(\mathsf{hk}_{\mathsf{meSSB}}) = Q$. We will show that $(\mathcal{P}_{\mathsf{nsKilian}}, \mathcal{V}_{\mathsf{nsKilian}})$ satisfies the properties in Definition 12. We will use the fact that $2^{\kappa^\epsilon} = 2^{((\log \Omega)^{1/\epsilon})^\epsilon} = \Omega$. In particular, a 2^{κ^ϵ}-hiding meSSB hash family is in fact $\Omega(n)$-hiding.

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

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

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

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

[12] In our case, with overwhelming probability over β_1, conditioned on the first three messages being $(\beta_1, \mathcal{P}^*(\beta_1), T(\beta_1))$, the remaining protocol is sound with probability 1.

where the last equality follows from Theorem 7 and the fact that the meSSB hash family is 2^{κ^ϵ}-hiding (which is $\Omega(n)$-hiding, as argued above). Furthermore, Corollary 7 gives that the reduction from the 2^{κ^ϵ}-hiding of the meSSB 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 $(\mathsf{hk}_{\mathsf{meSSB}}, Q)$ where $Q \leftarrow \mathcal{Q}_{\mathsf{nsPCP}}(1^\kappa)$ and $(\mathsf{hk}_{\mathsf{meSSB}}, \mathsf{td}_{\mathsf{meSSB}}) \leftarrow \mathsf{Gen}_{\mathsf{meSSB}}(1^\kappa, L, Q)$. Meanwhile, in the random case, the pair (β_1, β_2) is a pair $(\mathsf{hk}'_{\mathsf{meSSB}}, Q)$ where $Q, Q' \leftarrow \mathcal{Q}_{\mathsf{nsPCP}}(1^\kappa)$ and $(\mathsf{hk}'_{\mathsf{meSSB}}, \mathsf{td}'_{\mathsf{meSSB}}) \leftarrow \mathsf{Gen}_{\mathsf{meSSB}}(1^\kappa, L, Q')$. The Ω-indistinguishability of these two pairs follows from the $\Omega(n)$-index hiding property of the meSSB 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 $\mathsf{hk}_{\mathsf{meSSB}} \leftarrow \mathsf{Gen}_{\mathsf{meSSB}}(1^\kappa, L, Q)$ and $\mathsf{hk}_{\mathsf{meSSB}} \leftarrow \mathsf{Gen}_{\mathsf{meSSB}}(1^\kappa, L, Q')$ for an independent $Q' \leftarrow \mathcal{Q}_{\mathsf{nsPCP}}(1^\kappa)$, it feeds the pair $(Q, \mathsf{hk}_{\mathsf{meSSB}})$ to the distinguisher, and answers according to its response (without needing to use $\{\mathsf{td}_{\mathsf{meSSB},j}\}_{Q_j = Q'_j}$).

It follows from Theorem 8 that our instantiation of Kilian's protocol is 2^{κ^ϵ}-straight-line sound.

Theorem 10. *The protocol given in Figure 5 satisfies the following properties:*

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

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

– **Soundness:** *Assuming that the* meSSB *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 adaptively sound. In particular, for any $\mathrm{poly}(\Omega(n))$-size cheating prover $\mathcal{P}^*_{\mathsf{nsKilian}}$,*

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

Proof. Correctness is straightforward, and 2^{κ^ϵ}-straight-line soundness follows immediately from Theorem 8 and Lemma 2.

Recall that the eSSB hash family from Theorem 6 is sub-exponentially straight-line hiding assuming the sub-exponential hardness of LWE. Using this particular eSSB hash family in the construction of the meSSB hash family given in Figure 3 and using that the resulting meSSB hash family is 2^{κ^ϵ}-straight-line reducible from the 2^{κ^ϵ}-hiding of the underlying eSSB hash family, we obtain a meSSB hash family that is 2^{κ^ϵ}-straight-line reducible from the sub-exponential hardness of LWE. Combining this with the adaptive computational non-signaling PCPs given in Theorems 3 and 4, we obtain the following corollaries:

Corollary 2. *For any $\mathrm{poly}(n) \leq t \leq \exp(n)$, 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 adaptive t-computational non-signaling* PCP *for $\mathcal{L}_{\mathcal{U}}(t)$ from Theorem 3 and the* meSSB *hash family from Figure 5 with underlying* eSSB *hash family given in Theorem 6, is 2^{κ^ϵ}-straight-line (adaptive) sound. In particular, assuming the sub-exponential quantum hardness of* LWE*, this protocol is (adaptive) 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 3, 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 eSSB hash and opening is $\mathrm{poly}(\kappa) = \mathrm{polylog}(t)$, and the number of such eSSB 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 3. *For any* $\mathrm{poly}(n) \leq t \leq \exp(n)$ *and* $s = s(n) \geq \log t(n)$*, 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 adaptive* 2^s*-computational non-signaling* PCP *for* $\mathsf{NL}_{\mathcal{U}}(t, s)$ *from Theorem 4 and the* meSSB *hash family from Figure 5 with underlying* eSSB *hash family given in Theorem 6, is* 2^{κ^ϵ}*-straight-line (adaptively) sound. In particular, assuming the sub-exponential quantum hardness of* LWE*, this protocol is (adaptive) 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 4, 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 eSSB hash and opening is $\mathrm{poly}(\kappa) = \mathrm{polylog}(2^s) = \mathrm{poly}(s)$, and the number of such eSSB 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 $n \cdot \mathrm{poly}(s)$.

6 SNARG for Languages with Non-Signaling PCPs

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

We begin by defining BatchNP and SNARGs for BatchNP.

6.1 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 $\mathsf{L} \in \mathsf{NP}$.

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 13. *(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 14. *(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* 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* 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 15 (Σ-Soundness). *A SNARG $(\mathsf{Setup}_{\mathsf{L}^{\otimes N}}, \mathcal{P}_{\mathsf{L}^{\otimes N}}, \mathcal{V}_{\mathsf{L}^{\otimes N}})$ for $\mathsf{L}^{\otimes N} \in$ 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 11 ([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.*

6.2 SNARG for Languages with a Non-Signaling PCP

Suppose we have an adaptive Ω-computational non-signaling PCP $(\Pi, \mathcal{Q}_{\mathsf{nsPCP}}, \mathcal{V}_{\mathsf{nsPCP}})$ that is verifiable via tests (Definition 5) 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 3), 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 5.

At a high level, our SNARG for $\mathcal{L}$ works as follows: The honest prover first runs the BMW protocol on an adaptive computational non-signaling PCP with a meSSB hash function 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 meSSB hash family

$$(\mathsf{Gen}_{\mathsf{meSSB}}, \mathsf{Hash}_{\mathsf{meSSB}}, \mathsf{Open}_{\mathsf{meSSB}}, \mathsf{Verify}_{\mathsf{meSSB}}, \mathsf{Invert}_{\mathsf{meSSB}})$$

(see Construction 3).

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

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

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

$$m = \tau \cdot \log L + n + \ell_{\mathsf{meSSB},\mathsf{hk}} + \ell_{\mathsf{meSSB},\mathsf{hash}}. \tag{2}$$

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, $\langle Y \rangle = (x, \mathsf{hk}_{\mathsf{meSSB}}, \mathsf{rt})$, 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 $\langle Y \rangle$.

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 14.

Theorem 12. *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*

- *the* meSSB *hash family is* 2^{κ^ϵ}*-hiding,*
- *the* PCP *is adaptive* $n \leq \Omega = \Omega(n)$*-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 6) is* $\leq \Omega$*,*

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

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

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 $\Sigma(\lambda, m, N) = 2^{\ell_{\mathsf{meSSB},\mathsf{hash}}}$.

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

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

  It also samples

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

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

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

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

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

  (i.e. $\langle Y \rangle = (x, \mathsf{hk}_{\mathsf{meSSB}}, \mathsf{rt})$) 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 $\mathsf{o}_q = \mathsf{Open}_{\mathsf{meSSB}}(\mathsf{hk}_{\mathsf{meSSB}}, \pi, q)$. It outputs $\sigma = (\mathsf{rt}, \sigma_{\mathcal{M}^{\otimes N}})$.
- $\mathcal{V}_{\mathcal{L}}$ takes as input $\mathsf{crs} = (\mathsf{hk}_{\mathsf{meSSB}}, \mathsf{crs}_{\mathcal{M}^{\otimes N}})$, 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 $\langle Y \rangle = (x, \mathsf{hk}_{\mathsf{meSSB}}, \mathsf{rt})$.

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

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 $\mathcal{P}^*$ for which there is non-negligible δ such that

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

This is equal to

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

By Theorem 7 and the fact that a $2^{\kappa^\epsilon} = 2^{((\log \Omega)^{1/\epsilon})^\epsilon}$-hiding meSSB 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 meSSB hash family are binding on (used to generate $\mathsf{hk}_{\mathsf{meSSB}}$), and $\mathsf{td}_{\mathsf{meSSB}}$ is the trapdoor generated alongside $\mathsf{hk}_{\mathsf{meSSB}}$.

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

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

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

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 $\mathsf{hk}^*_{\mathsf{meSSB}}$ for which $\mathcal{P}^*(\mathsf{crs})$ outputs $(x, \mathsf{rt}, \sigma_{\mathcal{M}^{\otimes N}})$ with $x \notin \mathcal{L}$, $Y \notin \mathcal{M}^{\otimes N}$, 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 $\mathsf{crs} = (\mathsf{hk}^*_{\mathsf{meSSB}}, \mathsf{crs}_{\mathcal{M}^{\otimes N}})$ for some $\mathsf{crs}_{\mathcal{M}^{\otimes N}}$. Furthermore, there is some x^* and rt^* for which, with probability $\geq \frac{\delta'(\Omega)}{2^{n+\ell_{\mathsf{meSSB},\mathsf{hash}}}}$, this occurs and the x and rt output by $\mathcal{P}^*$ are equal to x^* and rt^*. In particular, for Y^* defined by $\langle Y^* \rangle = (x^*, \mathsf{hk}^*_{\mathsf{meSSB}}, \mathsf{rt}^*)$, we have that $Y^* \notin \mathcal{M}^{\otimes N}$.

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

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

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

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 $\mathsf{crs} = (\mathsf{hk}^*_{\mathsf{meSSB}}, \mathsf{crs}_{\mathcal{M}^{\otimes N}})$, to get x and $(\mathsf{rt}, \sigma_{\mathcal{M}^{\otimes N}})$. When the Merkle root rt that $\mathcal{P}^*$ output is equal to rt^* and x is equal to x^*, 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 meSSB hash family, which exists for some $\epsilon > 0$ assuming sub-exponential LWE (by Theorem 6 and Theorem 1),
- the adaptive 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 3 and 4, respectively,
- the 2^{λ^ϵ}-secure SNARG for $\mathcal{M}^{\otimes N}$ given in Theorem 4 which exists for some $\epsilon > 0$ assuming sub-exponential LWE, which means we may take $\lambda = (\ell_{\mathsf{meSSB},\mathsf{hash}})^{1/\epsilon}$ (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 $\Sigma(\lambda, m, N) = 2^{\lambda^\epsilon} = 2^{\ell_{\mathsf{meSSB},\mathsf{hash}}}$,

and taking $\epsilon > 0$ to be such that a 2^{κ^ϵ}-hiding meSSB 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 4. *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 adaptively 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 6 with the adaptive t-computational non-signaling PCP for $\mathcal{L}_{\mathcal{U}}(t)$ such that $N = \mathrm{poly}(t)$, which exists by Theorem 3, 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 3), 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 14). Note that $m = \tau \cdot \log L + n + \ell_{\mathsf{meSSB,hk}} + \ell_{\mathsf{meSSB,hash}} = \mathrm{poly}(n, \kappa, \log L) = \mathrm{poly}(n, \log t)$. 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. *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 $\mathsf{N}\mathcal{L}_{\mathcal{U}}(t, s)$ that is adaptively 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 6, instantiated with an adaptive 2^s-computational non-signaling PCP for $\mathsf{NTISP}(t, s)$ with $N = \mathrm{poly}(t)$ as given in Theorem 4, 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 $m = \tau \cdot \log L + n + \ell_{\mathsf{meSSB,hk}} + \ell_{\mathsf{meSSB,hash}}) = \mathrm{poly}(n, \kappa, \log L) = \mathrm{poly}(n, s, \log t) = \mathrm{poly}(n, s)$, 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 $\mathrm{poly}(\lambda, m, \log N) = \mathrm{poly}(n, s, \log t) = \mathrm{poly}(n, s)$.

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 13 ([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 $C(x) = 1$,*

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

– 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 14, 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 13. 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 13.

Our SNARG for $\mathcal{L}$ is described in Figure 7.

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

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

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

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

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 $\mathsf{crs} = (\mathsf{hk}_{\mathsf{meSSB}}, \mathsf{crs}_{\mathcal{M}^{\otimes N}}, \mathsf{crs}_{\mathsf{JKKZ}})$.

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

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

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

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

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 $\mathsf{o}_q = \mathsf{Open}_{\mathsf{meSSB}}(\mathsf{hk}_{\mathsf{meSSB}}, \pi, q)$. 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 $\mathsf{crs} = (\mathsf{hk}_{\mathsf{meSSB}}, \mathsf{crs}_{\mathcal{M}^{\otimes N}}, \mathsf{crs}_{\mathsf{JKKZ}})$, 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 $\langle Y\rangle = (x, \mathsf{hk}_{\mathsf{meSSB}}, \mathsf{rt})$.

Fig. 7. SNARG $(\mathsf{Setup}_{\mathcal{L}}, \mathcal{P}_{\mathcal{L}}, \mathcal{V}_{\mathcal{L}})(x)$ for $\mathcal{L}$

Theorem 14. *The algorithms $(\mathsf{Setup}_{\mathcal{L}}, \mathcal{P}_{\mathcal{L}}, \mathcal{V}_{\mathcal{L}})$ defined in Figure 7 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* meSSB *hash family is* $2^{\kappa^{\epsilon}}$*-hiding,*
- *the* PCP *is adaptive* $n \leq \Omega$*-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 7) 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* $2^{\eta^{\epsilon}}$*-soundness,*

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

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

For the sake of space, we omit the proof of Theorem 14.

Assuming sub-exponential LWE, there is some $\epsilon > 0$ such that both the following hold: a $2^{\kappa^{\epsilon}}$-hiding meSSB hash family exists and $(\mathsf{Setup}_{\mathsf{JKKZ}}, \mathcal{P}_{\mathsf{JKKZ}}, \mathcal{V}_{\mathsf{JKKZ}})$ has $2^{\eta^{\epsilon}}$-soundness. Assuming this, and assuming that there is a Σ-sound SNARG for $\mathcal{M}^{\otimes N}$ such that the verifier is a log-space uniform circuit of depth D, and using the adaptive computational non-signaling PCPs for $\mathcal{L}_{\mathcal{U}}(t)$ and $\mathsf{N}\mathcal{L}_{\mathcal{U}}(t, s)$ from Theorems 3 and 4, it follows that there exist SNARGs for $\mathcal{L}_{\mathcal{U}}(t)$ and $\mathsf{N}\mathcal{L}_{\mathcal{U}}(t, s)$ such that the prover runs in time $\mathrm{poly}(t)$, and the verifier runtime and communication complexity are $D \cdot \mathrm{poly}(n, \lambda, \log t)$ and $D \cdot \mathrm{poly}(n, \lambda, s)$ respectively. For the sake of space, we omit the formal statements of these results as well as the proof of Theorem 14.

Acknowledgements. We thank the anonymous TCC 2021 reviewers for their detailed and insightful comments.

References

Bar01. Boaz Barak. How to go beyond the black-box simulation barrier. In *FOCS*, pages 106–115, 2001. 5

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. 2, 4, 5

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. 6

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. 1, 5, 6, 9, 29, 30

BK18. Zvika Brakerski and Yael Tauman Kalai. Monotone batch np-delegation with applications to access control. *IACR Cryptology ePrint Archive*, 2018:375, 2018. 6

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. 1, 5, 6, 9

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. 5

BV11. Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. In *FOCS*, pages 97–106, 2011. 11

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. 4, 5

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. 4

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. 10

CJJ21. Arka Rai Choudhuri, Abhishek Jain, and Zhengzhong Jin. SNARGs for P from LWE. *IACR Cryptol. ePrint Arch.*, 2021. 2, 7, 21

CMSZ21. Alessandro Chiesa, Fermi Ma, Nicholas Spooner, and Mark Zhandry. Post-quantum succinct arguments. *IACR Cryptol. ePrint Arch.*, 2021:334, 2021. 3

CSW20. Ran Canetti, Pratik Sarkar, and Xiao Wang. Triply adaptive UC NIZK. *IACR Cryptol. ePrint Arch.*, 2020:1212, 2020. 3

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. 11

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. 1

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. 1

GK03. Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the fiat-shamir paradigm. In *FOCS*, pages 102–, 2003. 5

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. 5

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. 7

GKR08. Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In *STOC*, pages 113–122, 2008. 4

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. 5

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. 11

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. 4

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. 5

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. 1, 5, 11, 12, 13

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. 1, 6

JKKZ21. Ruta Jawale, Yael Tauman Kalai, Dakshita Khurana, and Rachel Zhang. Snargs for bounded depth computations and PPAD hardness from sub-exponential LWE. 2021. 5, 25

Kil92. Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *STOC*, pages 723–732, 1992. 1, 3

KO97. Eyal Kushilevitz and Rafail Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *FOCS*, pages 364–373, 1997. 10, 11

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. 1

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. 1, 29, 30

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. 1, 6, 9, 16

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. 4

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. 11

Mer87. Ralph C. Merkle. A digital signature based on a conventional encryption function. In *CRYPTO*, volume 293 of *Lecture Notes in Computer Science*, pages 369–378. Springer, 1987. 1

PR17. Omer Paneth and Guy N. Rothblum. On zero-testable homomorphic encryption and publicly verifiable non-interactive arguments. In *Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part II*, pages 283–315, 2017. 1

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. 5

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. 3

Wat09. John Watrous. Zero-knowledge against quantum attacks. *SIAM J. Comput.*, 39(1):25–58, 2009. 3

A Proof of Theorem 7

Theorem 7 shows the adaptive soundness of the BMW heuristic when applied to an adaptive computational non-signaling PCP and a meSSB hash family. The proof is nearly identical to that in [KRR13,BHK17] using an (adaptive) computational non-signaling PCP and a private information retrieval (PIR) scheme, and is provided here for completeness.

We first define the notion of an adaptive computational non-signaling PCP. For any ordered set[13] $U = (u_1, \ldots, u_\ell)$ and $J \subseteq [\ell]$, we let $U_J = (u_j)_{j \in J}$.

Definition 16 (Computational Non-Signaling Distributions). *A family of distributions $\{\mathcal{D}_Q\}_{Q \subset [L], |Q|=\ell}$ is* Ω-computational non-signaling *with locality ℓ if, for any $q_1, \ldots, q_\ell \in [L]$ and $q'_1, \ldots, q'_\ell \in [L]$, letting $J = \{j \in [\ell] : \ q_j = q'_j\}$, the following two distributions are Ω-indistinguishable (see Definition 1).*

- D_J *where* $D = (d_1, \ldots, d_\ell) \leftarrow \mathcal{D}_{(q_1, \ldots, q_\ell)}$,
- D'_J *where* $D' = (d'_1, \ldots, d'_\ell) \leftarrow \mathcal{D}_{(q'_1, \ldots, q'_\ell)}$

Definition 17 (Adaptive Computational Non-Signaling PCP). *An adaptive Ω-computational non-signaling* PCP *with locality ℓ is a* PCP $(\Pi, \mathcal{Q}_{\mathsf{PCP}}, \mathcal{V}_{\mathsf{PCP}})$ *where soundness holds against adaptive cheating provers mounting an Ω-non-signaling attack with locality ℓ. That is, for every Ω-computational non-signaling distribution $\{\mathcal{A}_S\}_{S \subset [L], |S| \leq \ell}$ with locality ℓ,*

$$\Pr\left[\mathcal{V}_{\mathsf{PCP}}(Q, x, A) = 1 \quad \wedge \quad x \notin \mathcal{L}\right] \leq 2^{-\kappa},$$

where the probability is over $(q_1, \ldots, q_\ell) \leftarrow \mathcal{Q}_{\mathsf{PCP}}(1^\kappa)$ and $(x, A) = (x, a_1, \ldots, a_\ell) \leftarrow \mathcal{A}_Q$, where $Q = (0, q_1, \ldots, q_\ell)$.[14]

Let $(\mathsf{Gen}_{\mathsf{meSSB}}, \mathsf{Hash}_{\mathsf{meSSB}}, \mathsf{Open}_{\mathsf{meSSB}}, \mathsf{Verify}_{\mathsf{meSSB}}, \mathsf{Open}_{\mathsf{meSSB}})$ be a meSSB hash family. We restate Theorem 7 below.

The BMW Protocol

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

- **Verifier:** $\mathcal{V}_{\mathsf{BMW}}$ computes PCP queries $(q_1, \ldots, q_\ell) \leftarrow \mathcal{Q}_{\mathsf{PCP}}(1^\kappa)$. He computes
$$(\mathsf{hk}_{\mathsf{meSSB}}, \mathsf{td}_{\mathsf{meSSB}}) \leftarrow \mathsf{Gen}_{\mathsf{meSSB}}(1^\kappa, L, \ell, (q_1, \ldots, q_\ell))$$
and sends $\mathsf{hk}_{\mathsf{meSSB}}$ to the prover.
- **Prover:** $\mathcal{P}_{\mathsf{BMW}}$ computes the PCP string $\pi = \Pi(x)$, and sends $\mathsf{rt} \leftarrow \mathsf{Hash}_{\mathsf{meSSB}}(\mathsf{hk}_{\mathsf{meSSB}}, \pi)$ to the verifier.
- **Verdict:** $\mathcal{V}_{\mathsf{BMW}}$ computes $(a_1, \ldots, a_\ell) \leftarrow \mathsf{Invert}_{\mathsf{meSSB}}([\ell], \mathsf{td}_{\mathsf{meSSB}}, \mathsf{rt})$ and accepts if and only if $\mathcal{V}_{\mathsf{PCP}}(x, (q_1, \ldots, q_\ell), (a_1, \ldots, a_\ell)) = 1$.

Fig. 8. BMW Heuristic with a meSSB Hash Function

Theorem 15 (Theorem 7, restated). *Let $(\Pi, \mathcal{Q}_{\mathsf{nsPCP}}, \mathcal{V}_{\mathsf{nsPCP}})$ be a* PCP *for a language $\mathcal{L}$ with adaptive $\Omega(n)$-computational non-signaling soundness and locality ℓ. Assume that the* meSSB *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 cheating prover $\mathcal{P}^*$ there is a negligible function μ such that*

$$\Pr\left[\mathcal{V}_{\mathsf{BMW}}(x, \mathsf{rt}, \mathsf{td}_{\mathsf{meSSB}}, (q_1, \ldots, q_\ell)) = 1 \ \wedge \ x \notin \mathcal{L}\right] \leq \mu(\Omega'),$$

[13] The works of [KRR13,BHK17] consider unordered sets. The analysis is nearly identical, however.

[14] We add the dummy 0 query because x is chosen adaptively depending on the PCP queries, and we think of it as the answer corresponding to this dummy query.

where $(x, \mathsf{rt}) = \mathcal{P}^*(\mathsf{hk}_{\mathsf{meSSB}})$ *and where the probability is over* $(q_1, \ldots, q_\ell) \leftarrow \mathcal{Q}_{\mathsf{PCP}}(1^\kappa)$ *and* $(\mathsf{hk}_{\mathsf{meSSB}}, \mathsf{td}_{\mathsf{meSSB}}) \leftarrow \mathsf{Gen}_{\mathsf{meSSB}}(1^\kappa, L, \ell, (q_1, \ldots, q_\ell))$. *Furthermore, the scheme is* Ω'*-straight-line sound.*

Proof. Suppose otherwise, that there is a $\mathrm{poly}(\Omega'(\kappa))$-size cheating prover $\mathcal{P}^*$ and a non-negligible function δ such that

$$\Pr\left[\mathcal{V}_{\mathsf{BMW}}(x, \mathsf{rt}, \mathsf{td}_{\mathsf{meSSB}}, (q_1, \ldots, q_\ell)) = 1 \ \wedge \ x \notin \mathcal{L}\right] \geq \delta(\Omega'),$$

where $(x, \mathsf{rt}) = \mathcal{P}^*(\mathsf{hk}_{\mathsf{meSSB}})$ and where the probability is over $(q_1, \ldots, q_\ell) \leftarrow \mathcal{Q}_{\mathsf{PCP}}(1^\kappa)$ and $(\mathsf{hk}_{\mathsf{meSSB}}, \mathsf{td}_{\mathsf{meSSB}}) \leftarrow \mathsf{Gen}_{\mathsf{meSSB}}(1^\kappa, L, \ell, (q_1, \ldots, q_\ell))$.

We will use $\mathcal{P}^*$ to construct an adaptive Ω-computational non-signaling strategy $\{\mathcal{A}_Q\}_{Q \subset [L], |Q| \leq \ell}$ such that

$$\Pr\left[\mathcal{V}_{\mathsf{PCP}}(Q, x, A) = 1 \quad \wedge \quad x \notin \mathcal{L}\right] \geq \delta(\Omega'), \tag{3}$$

where the probability is over $(q_1, \ldots, q_\ell) \leftarrow \mathcal{Q}_{\mathsf{PCP}}(1^\kappa)$ and $(x, A) = (x, a_1, \ldots, a_\ell) \leftarrow \mathcal{A}_Q$, where $Q = (0, q_1, \ldots, q_\ell)$. This would contradict the Ω-computational non-signaling soundness of the PCP.

Fix any $q_1, \ldots, q_\ell \in [L]$ and let $Q = (0, q_1, \ldots, q_\ell)$. The distribution $\mathcal{A}_Q$ is defined as follows:

1. Sample $(\mathsf{hk}_{\mathsf{meSSB}}, \mathsf{td}_{\mathsf{meSSB}}) \leftarrow \mathsf{Gen}_{\mathsf{meSSB}}(1^\kappa, L, \ell, (q_1, \ldots, q_\ell))$.
2. Compute $(x, \mathsf{rt}) = \mathcal{P}^*(\mathsf{hk}_{\mathsf{meSSB}})$.
3. Compute $A = (a_1, \ldots, a_\ell) = \mathsf{Invert}_{\mathsf{meSSB}}([\ell], \mathsf{td}_{\mathsf{meSSB}}, \mathsf{rt})$.
4. Output (x, A).

Our contradiction assumption immediately implies that Equation (3) holds. Thus it remains to argue that $\{\mathcal{A}_Q\}$ is a collection of Ω-computationally non-signaling distributions.

Fix any $q_1, \ldots, q_\ell \in [L]$ and $q'_1, \ldots, q'_\ell \in [L]$, and let $J = \{j \in [\ell] : \ q_j = q'_j\}$. Let

$$Q = (0, q_1, \ldots, q_\ell) \quad \text{and} \quad Q' = (0, q'_1, \ldots, q'_\ell),$$

let

$$(x, a_1, \ldots, a_\ell) \leftarrow \mathcal{A}_Q \quad \text{and} \quad (x', a'_1, \ldots, a'_\ell) \leftarrow \mathcal{A}_{Q'},$$

and let

$$A_J = (a_j)_{j \in J} \quad \text{and} \quad A'_J = (a'_j)_{j \in J}.$$

We need to prove that the distributions (x, A_J) and (x', A'_J) are Ω-indistinguishable.

Suppose otherwise, that there exists $q_1, \ldots, q_\ell \in [L]$, $q'_1, \ldots, q'_\ell \in [L]$ such that the corresponding distributions (x, A_J) and (x', A'_J) (as defined above) are not Ω-indistinguishable. Namely, there exists a $\mathrm{poly}(\Omega)$-size distinguisher D and a non-negligible function ϵ such that

$$|\Pr[D(x, A_J) = 1] - \Pr[D(x', A'_J) = 1]| \geq \epsilon(\Omega).$$

We will use this to break the Ω'-index hiding of the meSSB hash. An adversary for the Ω'-hiding of the meSSB hash picks the two sets of indices $i^0 = (q_1, \ldots, q_\ell)$ and $i^1 = (q'_1, \ldots, q'_\ell)$. Then, given $\mathsf{hk}_{\mathsf{meSSB}}$ generated by $(\mathsf{hk}_{\mathsf{meSSB}}, \mathsf{td}_{\mathsf{meSSB}}) \leftarrow \mathsf{Gen}_{\mathsf{meSSB}}(1^\kappa, L, \ell, i^{(b)})$ and trapdoor information $\mathsf{td}_{\mathsf{meSSB}}|_J$, does the following:

1. Compute $(x, \mathsf{rt}) = \mathcal{P}^*(\mathsf{hk}_{\mathsf{meSSB}})$.
2. Compute $A''_J = \mathsf{Invert}_{\mathsf{meSSB}}(J, \mathsf{td}_{\mathsf{meSSB}}|_J, \mathsf{rt})$.
3. Output $D(x, A''_J)$.

Note that the distinguishing advantage of this adversary is the same as the distinguishing advantage of the D, which is $\epsilon(\Omega)$. This contradicts the Ω-hiding of the meSSB hash family.