# Somewhere Statistical Soundness, Post-Quantum Security, and SNARGs for P

Yael Tauman Kalai* Microsoft Research

Vinod Vaikuntanathan† MIT

Rachel Yun Zhang‡ MIT

June 11, 2021

**Abstract**

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).

- First, we show that Kilian's protocol, instantiated with a computationally non-signaling PCP (Brakerski, Holmgren, and Kalai, STOC 2017) and a somewhere statistically binding hash family (Hubacek and Wichs, ITCS 2015), is an SSS argument.
- Secondly, we show that the soundness of SSS arguments can be proved in a straight-line manner, implying that they are also post-quantum sound if the underlying assumption is post-quantum secure. This provides a straightforward proof that Kilian's protocol, instantiated this way, is post-quantum sound under the post-quantum hardness of LWE (though we emphasize that a computationally non-signaling PCP exists only for deterministic languages, and more generally, for specific subclasses of non-deterministic languages such as NTISP, but not for all of NP).
- We put forward a natural conjecture that constant-round SSS arguments can be soundly converted into non-interactive arguments via the Fiat-Shamir transformation. We argue that SSS arguments evade the current Fiat-Shamir counterexamples, including the one for Kilian's protocol (Bartusek, Bronfman, Holmgren, Ma and Rothblum, TCC 2019) by requiring additional properties from both the hash family and the PCP.

As an additional result, we show that by using a computationally non-signaling PCP and a somewhere statistically binding hash family, one can efficiently convert any succinct non-interactive argument (SNARG) for BatchNP into a SNARG for P.

*E-mail:yael@microsoft.com

†E-mail:vinodv@csail.mit.edu

‡E-mail:rachelyz@mit.edu

# Contents

# 1 Introduction

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

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

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

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

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

## 1.1 Somewhere Statistically Sound Interactive Arguments

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

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

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

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 efficiently sampable and is computationally indistinguishable from a random pair $(\beta_1, \beta_2)$ of the verifier's first two messages. We emphasize that this in particular implies that the function $T$ has to be computationally inefficient.

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

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

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

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

3. Finally, we *conjecture* that any SSS interactive argument is *Fiat-Shamir friendly*; meaning that for any SSS interactive argument $(\mathcal{P}, \mathcal{V})$ there exists a hash family $\mathcal{H}$ such that applying the Fiat-Shamir paradigm w.r.t. $\mathcal{H}$ to $(\mathcal{P}, \mathcal{V})$ results with a sound non-interactive argument.

   We emphasize that we do not prove this claim, only conjecture it. We believe that it is a promising path for obtaining the first SNARG for all deterministic languages based on a standard post-quantum secure assumption. In particular, we conjecture that Kilian's protocol, with the instantiations above (SSB hash family and PCP with computational non-signaling soundness) is Fiat-Shamir friendly.

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

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

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

We believe an additional conceptual contribution of this work is our view of the first two messages of the Kilian's protocol, using an SSB hash function and a non-signaling PCP, as an instantiation of the 2-message protocol of Kalai, Raz and Rothblum [KRR14] (itself a provable version of an idea originally due to Biehl, Meyer and Wetzel [BMW99]). Using this view, we show how to construct a SNARG for P (and for NTISP) from any SNARG for BatchNP. We elaborate on this in Section 1.3.

### 1.1.1 SSS and Straight-Line Soundness

In a nutshell, the reason that any SSS protocol is post-quantum sound is due to the fact that it has *straight-line soundness*, meaning that a successfully cheating prover can be used in a black box way to break some complexity assumption.

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

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

The reduction $\mathcal{R}$ works as follows: It runs the cheating prover with $\beta_1$, and then upon recieving $\alpha_1 = \mathcal{P}^*(\beta_1)$, it sends $\mathcal{P}^*$ the challenge $\beta_2$. 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 $\beta_2$ is random), and otherwise it outputs $0$. By the assumption that $\mathcal{P}^*$ is convincing with non-negligible probability, if $\beta_1$ and $\beta_2$ 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 $\beta_2$ is random, as desired.

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

**Claim 1.2** (Informal). *Any* SSS *interactive argument that is proven sound under an assumption $A$ is also post-quantum sound if assumption $A$ holds w.r.t. quantum adversaries.*

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

### 1.1.2 SSS and Fiat-Shamir Friendliness

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

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

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

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

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

We note that Bartusek et al. [BBH+19] give an instantiation of Kilian's protocol for the trivial (empty) language for which applying the Fiat-Shamir paradigm provably results in a sound protocol. Their instantiation employs an SSB hash function and a particular PCP for the empty language, and the protocol is in fact SSS. Indeed, our conjecture is a stronger statement, namely that the notion of somewhere statistical soundness is sufficient to apply Fiat-Shamir soundly.

## 1.2 Instantiating an SSS version of Kilian

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

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

Hubáček and Wichs [HW15] constructed an SSB hash family assuming the hardness of LWE. This hash family is post-quantum secure assuming the post-quantum hardness of LWE. Moreover,

[KRR14, BHK17] constructed a PCP with computational non-signaling soundness for all deterministic languages and BatchNP languages, where for languages in DTIME$(t)$ the query complexity (or more precisely, the locality) is $\mathrm{polylog}(t)$, and for BatchNP languages it is $m \cdot \mathrm{polylog}(N)$, where $m$ is the length of a single witness and $N$ is the number of instances in the batch. These two results, together with Theorem 1.4, imply the following corollary.

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

More generally, we prove that if the PCP is $S$-computational non-signaling,[4] with locality $\ell$, and if we tree-commit to this PCP using $\ell$ hash functions that are SSB with $S$-security (i.e., even a $\mathrm{poly}(S)$-size adversary cannot break the SSB with probability non-negligible in $S$), then the resulting Kilian's protocol is SSS. We note that [BKK+18] constructed a PCP with $2^s$-computational non-signaling soundness (with locality $O(s)$) for any language in NTISP$(t, s)$. In addition, the SSB from [HW15] is $2^s$-secure (where the key size grows with $s$) assuming the sub-exponential hardness of LWE. We thus obtain the following corollary.

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

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

### 1.3 From BMW/KRR Back to Kilian

We take a somewhat anachronistic view and see Kilian's *four-message, public-coin* interactive argument as a natural interpolation of the Biehl-Meyer-Wetzel (BMW) heuristic. Recall that the BMW heuristic takes any PCP and any single-sever PIR scheme, and uses them to construct a two-message (succinct) argument where each PCP query is sent to the prover as a PIR query (see Section 2.4 for more details). The BMW heuristic is not known to be sound in general [DLN+04, DHRW16]; however, it is known to be computationally sound if the PCP is computational non-signaling [KRR13, BHK17]. The protocol is privately verifiable since the verifier needs to run the PIR decoding on the prover's message, in a sense decrypting it.

To see the relation to Kilian's protocol with an SSB hash and a non-signaling PCP, recall that an SSB hash family is a hash family $\mathcal{H}$ where each hash seed $s$ is associated with an index $i \in [N]$, where $N$ is the length of the input, such that $h_s(x)$ is statistically binding on $x_i$, and one can extract $x_i$ from the hash value $h_s(x)$ given a trapdoor $t$ that is generated together with $s$ (see Section 2.3, Definition 2.9). In our instantiation of Kilian's protocol, we hash the PCP with $\ell$ SSB

[4] We remark that for any $S_1 < S_2$, any PCP that is $S_1$-computational non-signaling is also $S_2$-computational non-signaling.

hash functions, where $\ell$ is the locality parameter of the PCP. Namely, the verifier's first message is $s_1, \ldots, s_\ell$ and the prover's respons is $(h_{s_1}(\pi), \ldots, h_{s_\ell}(\pi))$. By the semantic security property of a SSB hash family, and its inversion property given the trapdoor, this hash function can be thought of as a PIR scheme. Thus, these first two messages of Kilian's protocol are nothing but an instantiation of the BMW heuristic.

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

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

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

### 1.4 SNARGs: from BatchNP to P.

This view of the first two messages of the Kilian protocol as an instantiation of KRR, leads us to our final contribution, which is an alternative pathway to getting a SNARG for all of P. We show a reduction from constructing succinct non-interactive arguments (SNARGs) for P into the potentially simpler goal of constructing SNARGs for BatchNP. The starting point is the two-round preamble where the verifier sends the prover the description of an SSB hash function, and the verifier replies with the root of the Merkle hash of a non-signaling PCP. The key observation is that the remainder of the protocol can be a proof of the following BatchNP statement (which can be communicated in the first two rounds as well): for every possible query set $Q$ generated by the PCP verifier, there are values of $\pi_Q$ as well as openings $\mathsf{o}_Q$ such that (a) $(\pi_Q, \mathsf{o}_Q)$ constitutes a valid Merkle path; and (b) the PCP verifier accepts $(Q, \pi_Q)$.

We argue that this 2-message protocol is sound since if the instance being proven is false then by

the soundness of KRR the answers that are committed to by the hash root are rejecting, and hence by the somewhere-statistical binding property, the resulting BatchNP statement is false. Therefore, it seems that all we need to instantiate this approach is a SNARG for BatchNP.

There are several issues that come up in making this idea work. First, if the PCP has negligible soundness error, then the number of possible query sets generated by the verifier is super-polynomially large, meaning that the (honest) prover runtime is superpolynomial. Fortunately, all known PCP constructions (including the one from [KRR14]) 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.[5] Therefore, our BatchNP statement should rather be that for every test $\zeta$ there are values of $\pi_\zeta$ as well as openings $\mathsf{o}_\zeta$ such that (a) $(\pi_\zeta, \mathsf{o}_\zeta)$ constitutes a valid Merkle path; and (b) the PCP verifier accepts $(\zeta, \pi_\zeta)$. 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 P computation. Note that even though the proof length scales only poly-logarithmically with $N$, the *verifier runtime* scales at least linearly with $N$ since the verifier needs to at least read the entire statement. To solve this, we observe that in our case, the BatchNP statement actually has a succinct description. Thus, if there are succinct, easy to verify, proofs for succinctly specified BatchNP statements, we are back in business. We note that even if this is not the case, if the verifier's verdict function can be computed by a circuit that has depth only $\mathrm{polylog}(N)$ (but size $\mathrm{poly}(N)$), then again we are in business since we can use the SNARG for bounded depth computations (from sub-exponential LWE) [JKKZ20], and delegate this computation back to the prover.

Third and finally, note that the BatchNP proof system must have adaptive soundness since the prover gets to choose the BatchNP statement, in particular the Merkle root, *after* he recieves the CRS/first message of the BatchNP proof. Since the Merkle root 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 5.

# 2 Preliminaries

## 2.1 Straight-Line Reductions

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

**Definition 2.1.** *(Straight-Line Reductions) We say that an interactive argument $(\mathcal{P}, \mathcal{V})$ for a language $\mathcal{L} = \{\mathcal{L}_n\}_{n\in\mathbb{N}}$ is $t = t(n)$-*straight-line sound *if there is a* PPT *black box reduction $\mathcal{R}$ and a non-interactive $t$-decisional complexity assumption* [GK16],[6] *such that $\mathcal{R}$, given oracle access to any cheating prover $\mathcal{P}^*$ that breaks soundness with probability $1/\mathrm{poly}(t)$, interacts*

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

[6] We focus on decisional assumptions for simplicity, and because our reductions are from decisional assumptions.

*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* $t$-straight-line secure *(or $t$-secure via a straight-line reduction, or its security proof is straight line) if there is a* PPT *black box reduction $\mathcal{R}$ and a non-interactive $t$-decisional complexity assumption*[7] *such that $\mathcal{R}$, given oracle access to any adversary $\mathcal{A}$ that breaks the security of the primitive, interacts with $\mathcal{A}$* once *(without rewinding), and using the transcript obtained, breaks the assumption.*

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

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

## 2.2 Probabilistically Checkable Proofs

We recall the definition of a *probabilistically checkable proof* (PCP) and that of a *non-signaling* PCP. Intuitively, a PCP is a function $\Pi$ that takes as input a proof (for example, a witness $w$ for an NP statement $x$), and converts it into another proof $\pi = \Pi(x, w)$ which can be verified by a randomized verifier that reads only a few of its bits. For simplicity, in the rest of this paper we focus on deterministic laguages. We choose to do so since the landscape when considering non-deterministic languages is much more complicated, and we value simplicity and clarity over obtaining the most general results possible. That said, we refer the reader to Remark 2.6 about the non-deterministic setting.

In what follows we define the notion of PCP for deterministic languages (though traditionally PCPs are mainly considered for non-deterministic languages). For a function $t(\cdot)$, we consider the language $\mathcal{L}_{\mathcal{U}}(t) = \{\mathcal{L}_{\mathcal{U}}(t(n))\}_{n \in \mathbb{N}}$ 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 $(M, x)$ outputs 1 within $t(|(M, x)|)$ time steps.

In both the following definition and many other places in this paper, we will use another parameter $\kappa$, which will denote the security parameter used in various primitives. We will consider only the domain where $\kappa$ is relatively small compared to $t$, i.e. $\mathrm{poly}(\kappa) \leq t \leq \exp(\kappa)$. Eventually, we will take $\kappa = \mathrm{polylog}(t)$.

**Definition 2.3** (PCP)**.** *A probabilisitically checkable proof (*PCP*) for $\mathcal{L}_{\mathcal{U}}(t) = \{\mathcal{L}_{\mathcal{U}}(t(n))\}_{n \in \mathbb{N}}$ is a triple $(\Pi_t, \mathcal{Q}_{\mathsf{PCP},t}, \mathcal{V}_{\mathsf{PCP},t})$ with the following syntax:*

- *$\Pi_t$ is a deterministic algorithm that takes as input an instance $(M, x) \in \mathcal{L}_{\mathcal{U}}(t)$, runs in time polynomial in $t$, and outputs a proof string $\pi$. We denote by $L = |\pi|$.*
- *$\mathcal{Q}_{\mathsf{PCP},t}$ is a probabilistic query generation algorithm which takes as input a security parameter $1^\kappa$, runs time $\mathrm{poly}(\kappa, \log t)$, and generates a set of queries $q_1, \ldots, q_\ell \in [L]$.*

[7] It will be clear what the $t$-decisional complexity assumption is in each context.

- *$\mathcal{V}_{\mathsf{PCP},t}$ is a deterministic polynomial-time verification algorithm that takes as input an instance $(M,x)$, 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 $(M,x) \in \mathcal{L}_{\mathcal{U}}(t)$,*

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

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

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

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

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

**Remark 2.4.** *We also consider a stronger notion of* PCP *soundness known as* non-signaling soundness*, and more specifically* computational non-signaling soundness*. The precise definition is not needed in order to understand our result, since we use it in a black box way. We note that each such* PCP *is associated with a locality parameter $\ell$, which for simplicity can be thought of as the query complexity. We refer the reader to* [KRR14, BHK17] *for the precise definition.*

**Theorem 2.5** ([KRR14, BHK17])**.** *For any $\mathrm{poly}(n) \leq t(n) \leq \exp(n)$, there exists a 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, on input $(M,x)$, $(q_1,\ldots,q_\ell)$, and $(a_1,\ldots,a_\ell)$, $\mathsf{V}_{\mathsf{PCP},t}$ runs in time $|(M,x)| \cdot \mathrm{poly}(\ell)$.*

*Moreover, there exists a poly-time Turing machine $U$ such that each query set $Q = (q_1,\ldots,q_\ell) \in Q_{\mathsf{PCP},t}(1^\kappa)$ can be partitioned into $\theta = \kappa \cdot \mathrm{polylog}(t)$ tests $\zeta_1 \cup \ldots \cup \zeta_\theta$, 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 $U((M,x),Q|_{I_j},A|_{I_j}) = 1$ for every $j \in [\theta]$. There are a total of $\mathrm{poly}(t)$ many possible tests $\zeta$.*

**Remark 2.6.** *We note that computational non-signaling* PCP*s also exist for some* NP *languages. In particular [BHK17] constructed a computational non-signaling* PCP *for the class* BatchNP*. More specifically, for any* NP *language $\mathcal{L}$ they constructed a computational non-signaling* PCP *for $\mathcal{L}^{\otimes N}$ with locality $m \cdot \mathrm{polylog}(N)$, where $m$ is the length of a single witness for instances in $\mathcal{L}$, and $N$ is the number of* NP *instances in the batch. In addition, [BKK+18] constructed for each language in $\mathsf{NTISP}(t,s)$, the class of problems that can be solved non-deterministically in time $t$ and space $s$, a $2^s$-computational non-signaling* PCP *with locality $s \cdot \mathrm{polylog}(t)$. A $2^s$-computational non-signaling* PCP *is a* weaker *notion than a computational non-signaling* PCP*. More generally, for every $S_1 < S_2$, an $S_1$-computational non-signaling* PCP *is also an $S_2$-computational non-signaling* PCP*. Computational non-signaling is shorthand for $t$-computational non-signaling, where $\mathrm{poly}(t)$ is the runtime of the honest prover.*

## 2.3 Hash Function Families with Local Opening

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

**Definition 2.7** (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* $\kappa$ *in unary and an input length* $L$*, and outputs a hash key* $\mathsf{hk} \in \{0,1\}^{\ell_{\mathsf{hk}}}$.
- $\mathsf{Hash}(\mathsf{hk}, x)$ *takes as input a hash key* $\mathsf{hk} \in \{0,1\}^{\ell_{\mathsf{hk}}}$ *and an input* $x \in \{0,1\}^L$ *and outputs an element* $\mathsf{rt} \in \{0,1\}^{\ell_{\mathsf{hash}}}$.

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

**Definition 2.8** (Hash Family with Local Opening). *A* hash family with local opening *is a hash family* $(\mathsf{Gen}, \mathsf{Hash})$*, along with two additional* PPT *algorithms* $(\mathsf{Open}, \mathsf{Verify})$ *with the following syntax:*

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

*These algorithms should satisfy the property:*

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

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

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

In our construction, we will require hash functions with local openings that have an additional special property, that for any index $i \in [L]$ one can generate a hash key such that a hash value is statistically binding at position $i$: namely, all the preimages of a hashed value have the same value at position $i$. These hash functions are called *somewhere statistically binding* and were first defined in [HW15]. We will also require that the statistically bound value at position $i$ can be recovered via an invert function that is efficient with the aid of a trapdoor.[8]

**Definition 2.9** (SSB Hash Family). *A* $S = S(\kappa)$-hiding somewhere statistically binding (SSB) hash family *is a hash family with local opening* $(\mathsf{Gen}, \mathsf{Hash}, \mathsf{Open}, \mathsf{Verify})$*, where*

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

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

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

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

*These algorithms should satisfy the following properties:*

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

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

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

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

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

- **Somewhere Statistically Binding:**[9] *For any* $\kappa \in \mathbb{N}$*,* $L \leq 2^\kappa$*,* $i \in [L]$ *and* $\mathsf{rt} \in \{0,1\}^{\ell_{\mathsf{hash}}}$*,*

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

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

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

**Theorem 2.10** ([HW15])**.** *Assuming the sub-exponential hardness of the learning with errors (*LWE*) problem, there exists a* $2^{\kappa^\epsilon}$*-hiding* SSB *hash family for some* $\epsilon > 0$*. The* $2^{\kappa^\epsilon}$*-hiding is via a straight-line reduction (see Definition 2.1).*

## 2.4 Converting a PCP into a Succinct Interactive Argument

In this section, we review two known methods for converting a PCP into a succinct argument. The first is Kilian's protocol [Kil92], which results in a 4-message, publicly verifiable succinct argument. The second is the BMW protocol [BMW99], which results in a 2-message, privately verifiable succinct argument.

**Kilian's Protocol.** Kilian's transformation uses a hash family with local opening and a PCP scheme to construct a 4-round succinct argument, given in Figure 1.

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

**Kilian's Protocol**

Fix any hash family with local opening $\mathcal{H} = (\mathsf{Gen}, \mathsf{Hash}, \mathsf{Open}, \mathsf{Verify})$ and a PCP scheme $(\Pi_t, \mathcal{Q}_{\mathsf{PCP},t}, \mathcal{V}_{\mathsf{PCP},t})$ for the language $\mathcal{L}_{\mathcal{U}}(t) = \{\mathcal{L}_{\mathcal{U}}(t(n))\}_{n\in\mathbb{N}}$. Denote the length of a PCP proof by $L = L(n)$. On input $(M, x) \in \mathcal{L}_{\mathcal{U}}(t)$ and security parameter $1^\kappa$, 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 $\mathsf{hk}$ to the prover.
- **First prover's meessage:** $\mathcal{P}_{\mathsf{Kilian}}$ computes the PCP proof $\pi = \Pi_t(M, x)$, and its hash value $\mathsf{rt} = \mathsf{Hash}(\mathsf{hk}, \pi)$. It sends $\mathsf{rt}$ to the verifier.
- **Second verifier's message:** $\mathcal{V}_{\mathsf{Kilian}}$ computes a set of queries $(q_1, \ldots, q_\ell) \leftarrow \mathcal{Q}_{\mathsf{PCP},t}(1^\kappa)$, 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 $\mathcal{V}_{\mathsf{PCP},t}((M, x), (q_1, \ldots, q_\ell), (\pi_{q_1}, \ldots, \pi_{q_\ell})) = 1$ and for every $i \in [\ell]$, $\mathsf{Verify}(\mathsf{hk}, \mathsf{rt}, q_i, \pi_{q_i}, \mathsf{o}_i) = 1$.

Figure 1: Kilian's Protocol $(\mathcal{P}_{\mathsf{Kilian}}, \mathcal{V}_{\mathsf{Kilian}})$ for $\mathcal{L}_{\mathcal{U}}(t)$

**The BMW Heuristic.** The BMW heuristic allows one to query a PCP proof using a private information retrieval (PIR) scheme, which we define below.

**Definition 2.11** ([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* $\kappa$ *in unary, an input size* $L$*, and an index* $q \in [L]$*, and outputs a query* $\hat{q}$ *along with a trapdoor* $\mathsf{td}$*.*
- $\mathsf{Answer}(\hat{q}, x)$ *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* $\mathsf{td}$ *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 $\mu$ 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 which was followed up by several other works [GR05, Lip05, BV11, DGI+19].

**Theorem 2.12** ([BV11, DGI+19]). *For any function $S : \mathbb{N} \to \mathbb{N}$, there exists a $S$-private 1-server* PIR *scheme with* $\mathrm{polylog}(L)$ *query complexity for length-$L$ databases, under the $S$-hardness of the* LWE*, Quadratic Residuosity, or* DDH *assumptions. Moreover, these schemes are $S$-straight-line secure (see Defintion 2.1).*

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

**The BMW Protocol**

Let $(\Pi_t, Q_{\mathsf{PCP},t}, \mathsf{V}_{\mathsf{PCP},t})$ be a PCP for $\mathcal{L}_{\mathcal{U}}(t)$, and let $\kappa$ be such that $\mathrm{poly}(\kappa) \leq t \leq \exp(\kappa)$. On input $(M, x)$, the 2-message protocol $(\mathcal{P}_{\mathsf{BMW}}, \mathcal{V}_{\mathsf{BMW}})$ proceeds as follows:

- **Verifier:** $\mathcal{V}_{\mathsf{BMW}}$ computes $(q_1, \ldots, q_\ell) \leftarrow \mathcal{Q}_{\mathsf{PCP},t}(1^\kappa)$. 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 $\pi = \Pi_t(M, x)$, 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 $V_{\mathsf{PCP},t}((M, x), (q_1, \ldots, q_\ell), (a_1, \ldots, a_\ell)) = 1$.

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

**Theorem 2.13** ([KRR14, BHK17]). *The protocol $(\mathcal{P}_{\mathsf{BMW}}, \mathcal{V}_{\mathsf{BMW}})$ has the following guarantees for $\mathrm{poly}(n) \leq t = t(n) \leq \exp(n)$:*

- **Completeness:** *For every $(M, x) \in \mathcal{L}_{\mathcal{U}}(t)$,*

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

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

- **Straight-Line Soundness:** *Suppose the underlying* PCP $(\Pi_t, \mathcal{Q}_{\mathsf{PCP},t}, \mathcal{V}_{\mathsf{PCP},t})$ *has computational non-signaling soundness (see Remark 2.4), and assume that* $2^{-\kappa} = \mathrm{negl}(t)$. *Let* $t'$ *be such that* $t'(\kappa) = t$. *Then, assuming the* PIR *scheme has* $t'$*-privacy, for every* $\mathrm{poly}(t)$*-size cheating prover* $\mathcal{P}^*$, *there exists a negligible function* $\mu$ *such that for every* $(M, x) \notin \mathcal{L}_{\mathcal{U}}(t)$,

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

*Moreover, the soundness is* $t'$*-straight-line (Definition 2.1).*[10]

**Remark 2.14.** *More generally, as we mentioned in Remark 2.6, [BHK17] constructed a computational non-signaling* PCP *for the class* BatchNP *with locality* $\ell = m \cdot \mathrm{polylog}(N)$, *where* $m$ *is the length of a single witness and* $N$ *is the number of instances in the batch. As a corollary they concluded that the* BMW *heuristic can be instantiated securely for this class, where the communication complexity grows with the locality* $\ell$.[11] *In addition, [BKK+18] constructed a* $2^s$*-computational non-signaling* PCP *for the class* $\mathsf{NTISP}(t, s)$ *with locality* $s$. *As a corollary, they concluded that the* BMW *heuristic can be instantiated securely for this class if the underlying* PIR *scheme is* $2^s$*-private.*[12] *We note that both proofs are* $t$*-straight-line sound, where* $t$ *is the running time of the honest prover (given the witnesses) and where the claim is that a* $\mathrm{poly}(t)$*-size cheating prover cannot break soundness with probability better than* $\mathrm{negl}(t)$.

# 3 Somewhere Statistically Sound Interactive Arguments

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

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

**Definition 3.2.** *An interactive argument* $(\mathcal{P}, \mathcal{V})$ *for a language* $\mathcal{L} = \{\mathcal{L}_n\}_{n \in \mathbb{N}}$ *is* $t = t(n)$-somewhere statistically sound (SSS) *if for every first verifier message* $\beta_1$, *there exists a second verifier message* $T(\beta_1)$ *such that:*

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

  *Moreover, this condition holds in a straight-line manner (Definition 2.1); i.e., there is a black box reduction* $\mathcal{R}$ *and a* $t$*-decisional complexity assumption such that* $\mathcal{R}$, *given oracle access to a* $\mathrm{poly}(t)$*-size cheating prover* $\mathcal{P}^*$ *that finds* $\mathcal{P}^*(\beta_1)$ *for which the protocol beginning with* $(\beta_1, \mathcal{P}^*(\beta_1), T(\beta_1)$ *is not* $t$*-statistically sound, sends this prover a single query* $\beta_1$, *and uses the response* $\mathcal{P}^*(\beta_1)$ *to break the assumption.*

[10] Note that the $t'$-privacy of the PIR scheme is a $t'$-decisional assumption. Moreover, if we use one of the PIR schemes from Theorem 2.12 then it further reduces (via a striaght-line reduction) to a standard $t'$-decisional assumption, such as $t'$-hardness of LWE, QR, or DDH.

[11] We mention that to obtain adaptive soundness they need to rely on a PIR scheme that is $2^m$-private. This is in contrast to $\mathcal{L}_{\mathcal{U}}$ for which adaptive soundness can be obtained assuming $t'$-privacy of the PIR scheme.

[12] They also obtained adaptive soundness under this assumption.

- **Efficient Sampleability:** *The distribution* $(\beta_1, T(\beta_1))$*, for a random* $\beta_1$*, is efficiently sampleable.*
- **Computational Indistinguishability:** *For any* $\mathrm{poly}(t)$*-size distinguisher* $\mathcal{D}$*,*

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

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

**Theorem 3.3.** *Any* $t$-SSS *interactive argument* $(\mathcal{P}, \mathcal{V})$ *is* $t$*-straight-line sound.*

*Proof.* To prove straight-line soundness, we will define a straight-line reduction from the somewhere statistically sound and computational indistinguishability assumptions to the $t$-soundness of $(\mathcal{P}, \mathcal{V})$. Then, combining with the fact that there is a straight-line reduction from some $t$-decisional complexity assumption to the somewhere statistically sound property (and using the fact that the computational indistinguishability property is a $t$-decisional complexity assumption since pairs $(\beta_1, T(\beta_1))$ are efficiently sampleable), we obtain that there is a straight-line reduction from the base assumptions to the $t$-soundness of $(\mathcal{P}, \mathcal{V})$.

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

Note that

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

so the distinguishing advantage of the reduction is

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

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

# 4 Kilian's Protocol is Somewhere Statistically Sound

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

### 4.1 The BMW Heuristic with SSB Hash Families

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

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

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

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

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

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

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

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

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

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

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

**An $\ell$-SSB Hash Family**

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

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

is defined as follows:

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

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

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

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

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

Note that if we instantiate the Kilian protocol with an $\ell$-SSB hash family and a PCP with computational non-signaling soundness, then by Theorem 2.13 and Claim 4.1, the statistically committed answers will be rejecting with high probability. This is summarized in the following corollary.

**Corollary 4.3** (Corollary of Theorem 2.13)**.** *For $\mathrm{poly}(n) \leq t = t(n) \leq \exp(n)$, let $(\Pi_t, \mathcal{Q}_{\mathsf{nsPCP},t}, \mathcal{V}_{\mathsf{nsPCP},t})$ be a* PCP *for $\mathcal{L}_{\mathcal{U}}(t)$ with computational non-signaling soundness, and let $\mathcal{H}_{\ell\text{-}\mathsf{SSB}}$ be the $\ell$-*SSB *hash family given in Figure 3, where $\ell$ is the size of a* PCP *query generated by $\mathcal{Q}_{\mathsf{nsPCP},t}(1^\kappa)$. Assume that $\kappa$ satisfies $2^{-\kappa} = \mathrm{negl}(t)$ and that the underlying* SSB *hash family instantiated with security parameter $\kappa$ is $t'$-hiding, where $t'$ is such that $t'(\kappa) = t$.*

*Then for any* $\mathrm{poly}(t)$*-size adversary* $\mathcal{A}$ *and for any* $(M,x) \notin \mathcal{L}_{\mathcal{U}}(t)$*,*

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

*Moreover, this is proven via a straight-line reduction (Definition 2.1).*[13]

### 4.2 Kilian with No-Signaling PCP and $\ell$-SSB Hashing

We instantiate Kilian's protocol with two ingredients: a computational non-signaling PCP $(\Pi_t, \mathcal{Q}_{\mathsf{nsPCP},t}, \mathcal{V}_{\mathsf{nsPCP},t})$ for $\mathcal{L}_{\mathcal{U}}(t)$ as given in Theorem 2.5, and the $\ell$-SSB hash family given in Figure 3. The resulting protocol is described in Figure 4.

**Kilian's protocol instantiated with a computational non-signaling PCP and an $\ell$-SSB hash family**

Let $\epsilon \in (0,1)$ be a small constant, and let $\kappa = \kappa(n) = (\log t(n))^{2/\epsilon}$. Let $(\Pi_t, \mathcal{Q}_{\mathsf{nsPCP},t}, \mathcal{V}_{\mathsf{nsPCP},t})$ be a computational non-signaling PCP for $\mathcal{L}_{\mathcal{U}}(t)$, and let $(\mathsf{Gen}_{\ell\text{-}\mathsf{SSB}}, \mathsf{Hash}_{\ell\text{-}\mathsf{SSB}}, \mathsf{Open}_{\ell\text{-}\mathsf{SSB}}, \mathsf{Verify}_{\ell\text{-}\mathsf{SSB}}, \mathsf{Invert}_{\ell\text{-}\mathsf{SSB}})$ be a $\ell$-SSB hash family. On input $(M,x)$, the 4-message protocol $(\mathcal{P}_{\mathsf{nsKilian}}, \mathcal{V}_{\mathsf{nsKilian}})$ proceeds as follows.

- **First verifier's message:** $\mathcal{V}_{\mathsf{nsKilian}}$ samples $Q \leftarrow \mathcal{Q}_{\mathsf{nsPCP},t}(1^\kappa)$ and $(\mathsf{hk}_{\ell\text{-}\mathsf{SSB}}, \mathsf{td}_{\ell\text{-}\mathsf{SSB}}) \leftarrow \mathsf{Gen}_{\ell\text{-}\mathsf{SSB}}(1^\kappa, L, Q)$, and sends $\mathsf{hk}_{\ell\text{-}\mathsf{SSB}}$ to the prover.
- **First prover's meessage:** $\mathcal{P}_{\mathsf{nsKilian}}$ computes the PCP proof $\pi = \Pi_t(M,x)$ and its hash value $\mathsf{rt} = \mathsf{Hash}_{\ell\text{-}\mathsf{SSB}}(\mathsf{hk}_{\ell\text{-}\mathsf{SSB}}, \pi)$. It sends $\mathsf{rt}$ to the verifier.
- **Second verifier's message:** $\mathcal{V}_{\mathsf{nsKilian}}$ computes a set of queries $(q_1,\ldots,q_\ell) \leftarrow \mathcal{Q}_{\mathsf{nsPCP},t}(1^\kappa)$, and sends $(q_1,\ldots,q_\ell)$ to the prover.
- **Second prover's message:** $\mathcal{P}_{\mathsf{nsKilian}}$ computes for every $i \in [\ell]$ the opening $\mathsf{o}_i = \mathsf{Open}_{\ell\text{-}\mathsf{SSB}}(\mathsf{hk}_{\ell\text{-}\mathsf{SSB}}, \pi, q_i)$, and sends $\{\pi_{q_i}, \mathsf{o}_i\}_{i\in[\ell]}$ to the verifier.
- **Verdict:** $\mathcal{V}_{\mathsf{nsKilian}}$ accepts if and only if $\mathcal{V}_{\mathsf{nsPCP},t}((M,x),(q_1,\ldots,q_\ell),(\pi_{q_1},\ldots,\pi_{q_\ell})) = 1$ and for every $i \in [\ell]$, $\mathsf{Verify}_{\ell\text{-}\mathsf{SSB}}(\mathsf{hk}_{\ell\text{-}\mathsf{SSB}}, \mathsf{rt}, q_i, \pi_{q_i}, \mathsf{o}_i) = 1$.

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

With these ingredients, the resulting Kilian's protocol is an SSS argument, as we show below.

**Lemma 4.4.** *Let* $\mathrm{poly}(n) \le t = t(n) \le \exp(n)$*. Assuming the underlying* SSB *hash family is* $2^{\kappa^\epsilon}$*-hiding,* $(\mathcal{P}_{\mathsf{nsKilian}}, \mathcal{V}_{\mathsf{nsKilian}})$ *is a* $t$*-*SSS *interactive argument. Moreover, the somewhere statistical soundness and computational indistinguishability properties (Definition 3.2) are* $t$*-straight-line secure, i.e. there is a black box reduction from the* $2^{\kappa^\epsilon}$*-hiding of the* SSB *hash family to the two properties.*

[13] Note that the $S$-Index-Hiding property of the SSB is a non-interactive $S$-decisional assumption.

*Proof.* For $(\mathsf{hk}_{\ell\text{-SSB}}, \mathsf{td}_{\ell\text{-SSB}}) \leftarrow \mathsf{Gen}_{\ell\text{-SSB}}(1^\kappa, L, Q)$, define $T(\mathsf{hk}_{\ell\text{-SSB}}) = Q$. We will show that $(\mathcal{P}_{\mathsf{nsKilian}}, \mathcal{V}_{\mathsf{nsKilian}})$ satisfies the properties in Definition 3.2. We will use the fact that $2^{\kappa^\epsilon} = 2^{((\log t)^{2/\epsilon})^\epsilon} = t^{\log t} = \omega(t)$. In particular, the $\ell$-SSB hash family is $2^{\kappa^\epsilon}$-index-hiding by Lemma 4.2 (using that $\ell = |Q| = \kappa \cdot \mathrm{polylog}(t) = \mathrm{polylog}(t) = o(2^{\kappa^\epsilon})$ from Definition 2.3), which means that it is index hiding against $\mathrm{poly}(t(n))$-size adversaries (with probability $\mathrm{negl}(t(n))$).

- **Somewhere Statistically Sound:** The somewhere statistically sound property of Definition 3.2 is equivalent to the condition that for every $\mathrm{poly}(t)$-size $\mathcal{P}^* = (\mathcal{P}_1^*, \mathcal{P}_2^*)$,

$$\Pr\left[\begin{array}{r|r} \mathcal{V}_{\mathsf{nsPCP},t}((M,x), Q, (a_1, \ldots, a_\ell)) = 1 & Q \leftarrow \mathcal{Q}_{\mathsf{nsPCP},t}(1^\kappa) \\ \wedge\ \mathsf{Verify}_{\ell\text{-SSB}}(\mathsf{hk}_{\ell\text{-SSB}}, \mathsf{rt}, q_j, a_j, \mathsf{o}_j) = 1\ \forall j \in [\ell] & (\mathsf{hk}_{\ell\text{-SSB}}, \mathsf{td}_{\ell\text{-SSB}}) \leftarrow \mathsf{Gen}_{\ell\text{-SSB}}(1^\kappa, L, Q) \\ & (\mathsf{rt}, \mathsf{state}) \leftarrow \mathcal{P}_1^*(\mathsf{hk}_{\ell\text{-SSB}}) \\ & \{a_j, \mathsf{o}_j\}_{j\in[\ell]} \leftarrow \mathcal{P}_2^*(Q, \mathsf{state}) \end{array}\right]$$

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

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

  where the last equality follows from Corollary 4.3 and the fact that the $\ell$-SSB hash family is $2^{\kappa^\epsilon}$-hiding (which is $t(n)$-index hiding, as argued above). Furthermore, the reduction from the $2^{\kappa^\epsilon}$-hiding of the $\ell$-SSB hash family to the somewhere statistical soundness is black box, i.e. the proof is straight-line.

- **Efficient Sampability:** A pair $(\beta_1, T(\beta_1))$ can be sampled in the following manner: first sample $Q \leftarrow \mathcal{Q}_{\mathsf{nsPCP},t}(1^\kappa)$, and then sample $(\mathsf{hk}_{\ell\text{-SSB}}, \mathsf{td}_{\ell\text{-SSB}}) \leftarrow \mathsf{Gen}_{\ell\text{-SSB}}(1^\kappa, L, Q)$. Set $\beta_1 = \mathsf{hk}_{\ell\text{-SSB}}$ and $T(\beta_1) = Q$. This can be done in $\mathrm{poly}(\kappa, \log t) + \mathrm{poly}(\kappa, \log L, \ell) = \mathrm{polylog}(t) \leq \mathrm{poly}(n)$ time.

- **Computational Indistinguishability:** In the formatted case, the pair $(\beta_1, T(\beta_1))$ is a pair $(\mathsf{hk}_{\ell\text{-SSB}}, Q)$ where $Q \leftarrow \mathcal{Q}_{\mathsf{nsPCP},t}(1^\kappa)$ and $(\mathsf{hk}_{\ell\text{-SSB}}, \mathsf{td}_{\ell\text{-SSB}}) \leftarrow \mathsf{Gen}_{\ell\text{-SSB}}(1^\kappa, L, Q)$. Meanwhile, in the random case, the pair $(\beta_1, \beta_2)$ is a pair $(\mathsf{hk}'_{\ell\text{-SSB}}, Q)$ where $Q, Q' \leftarrow \mathcal{Q}_{\mathsf{nsPCP},t}(1^\kappa)$ and $(\mathsf{hk}'_{\ell\text{-SSB}}, \mathsf{td}'_{\ell\text{-SSB}}) \leftarrow \mathsf{Gen}_{\ell\text{-SSB}}(1^\kappa, L, Q')$. The indistinguishability of these two pairs for $\mathrm{poly}(t)$-size distinguishers (and with probability non-negligible in $t$) follows from the $t(n)$-index hiding property of the $\ell$-SSB hash family via a black box reduction: The reduction picks $Q \leftarrow \mathcal{Q}_{\mathsf{nsPCP},t}(1^\kappa)$ at random. Then, to distinguish between $\mathsf{hk}_{\ell\text{-SSB}} \leftarrow \mathsf{Gen}_{\ell\text{-SSB}}(1^\kappa, L, Q)$ and $\mathsf{hk}_{\ell\text{-SSB}} \leftarrow \mathsf{Gen}_{\ell\text{-SSB}}(1^\kappa, L, Q')$ for an independent $Q' \leftarrow \mathcal{Q}_{\mathsf{nsPCP},t}(1^\kappa)$, it feeds the pair $(Q, \mathsf{hk}_{\ell\text{-SSB}})$ to the distinguisher, and answers according to its response.

□

It follows from Theorem 3.3 that our instantiation of Kilian's protocol is straight-line sound.

**Theorem 4.5.** *For $\mathrm{poly}(n) \leq t = t(n) \leq \exp(n)$, the protocol given in Figure 4 satisfies the following properties:*

- **Correctness:** *For any* $(M, x) \in \mathcal{L}_{\mathcal{U}}(t)$ *and* $\epsilon > 0$*,*

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

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

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

- **Runtimes:** *The prover runs in time* $\mathrm{poly}(t)$*. The verifier runs in time* $n \cdot \mathrm{polylog}(t)$*. The communication complexity is* $\mathrm{polylog}(t)$*.*

*Proof.* Correctness is straightforward, and $t$-straight-line soundness follows immediately from Theorem 3.3 and Lemma 4.4. The complexity claims follow from the following points:

- By Theorem 2.5, the size of the PCP proof is $\mathrm{poly}(t)$, so $\mathcal{P}_{\mathsf{nsKilian}}$ can compute the hash and openings in time $\mathrm{poly}(t)$.
- The size of a single SSB hash and opening is $\mathrm{poly}(\kappa) = \mathrm{polylog}(t)$, and the number of such SSB hashes and openings is $\ell = \kappa \cdot \mathrm{polylog}(t) = \mathrm{polylog}(t)$, for a total communication complexity of $\mathrm{polylog}(t)$.
- The verifier can check that all the answers and openings are consistent with rt in time $\mathrm{polylog}(t)$. He also runs $\mathcal{V}_{\mathsf{nsPCP},t}$, which takes time $n \cdot \mathrm{poly}(\ell) = n \cdot \mathrm{polylog}(t)$, for a total verifier runtime of $n \cdot \mathrm{polylog}(t)$.

□

Recall that the SSB hash family from Theorem 2.10 is sub-exponentially straight-line hiding assuming the sub-exponential hardness of LWE. Using this particular SSB hash family in the $\ell$-SSB hash family, we obtain the following corollary:

**Corollary 4.6.** *Using the* SSB *hash family from Theorem 2.10, the argument* $(\mathcal{P}^*_{\mathsf{nsKilian}}, \mathcal{V}^*_{\mathsf{nsKilian}})$ *for* $\mathcal{L}_{\mathcal{U}}(t)$ *is* $t$*-straight-line sound for some* $\epsilon > 0$ *assuming the sub-exponential hardness of* LWE*.*

# 5 SNARG for P

In this section, we present the construction of a SNARG for P, assuming the existence of a SNARG for BatchNP. Essentially, the honest prover in our SNARG first runs the BMW protocol on a computationally no-signaling PCP with SSB hash functions to produce a short commitment rt to the entire PCP. She then provides a short proof 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, as the answers and openings serve as the witness; now the claim that *all possible* verifier tests have accepting answers and openings is a BatchNP statement. We begin by defining BatchNP.

### 5.1 BatchNP

For an NP relation $R$ with corresponding language $L$, define

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

and

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

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

In our application, we will be interested in the case where $N$ is much larger than $m$, the size of a single instance $x_i$. In order for the verifier to digest the BatchNP instance, we will require that there is a succinct representation of the instance.

**Definition 5.1.** *(Succinct Description of a Set) A set $S \subseteq \{0,1\}^m$ 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 $S$.*

**SNARGs for** BatchNP**.** Our SNARG for P relies on the existence of a SNARG for BatchNP, which we define below. In our definition, we make the additional requirement that the verifier is super-efficient, i.e., runs in time $\mathrm{poly}(m, \log N)$, if the BatchNP statement can be succinctly described using $\mathrm{poly}(m, \log N)$ bits. This differs from the traditional notion of a SNARG for BatchNP, which allows the verifier enough time to read the entire BatchNP instance (which is of size $N \cdot m$), and only requires that the proof string sent over is very short.

We remark that this succinctness condition is not needed, and all we need is that the verifier's verdict function can be computed by a circuit of $\mathrm{poly}(m, \log N)$ depth. In this case, the verifier's verdict function can be delegated by using the recent SNARG for bounded depth computations from sub-exponential LWE [JKKZ21].

**Definition 5.2.** *(SNARG for* BatchNP*) A SNARG for a language $L \in$ BatchNP with corresponding relation $R$ is a tuple of* PPT *algorithms $(\mathsf{Setup}_L, \mathcal{P}_L, \mathcal{V}_L)$ with the following syntax:*

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

*These algorithms should satisfy the following properties:*

- **Correctness:** *If $(X, W) \in R$, then*

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

- **$S$-Soundness:** *For all $\lambda, m, N$ and all $\mathrm{poly}(S(\lambda, m, N))$-size cheating prover $\mathcal{P}^*_L$, and for any $X \notin L$,*

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

## 5.2 SNARG for P

In what follows, rather than constructing a SNARG only for languages in P, we construct a SNARG for $\mathcal{L}_{\mathcal{U}}(t)$, for every $\mathrm{poly}(n) \le t \le \exp(n)$. To this end, let $(\Pi_t, \mathcal{Q}_{\mathsf{nsPCP},t}, \mathcal{V}_{\mathsf{nsPCP},t})$ be the computational non-signaling PCP for $\mathcal{L}_{\mathcal{U}}(t)$ from Theorem 2.5. Let $L$ be the size of the PCP and $\ell$ be the locality. Let $N = \mathrm{poly}(t)$ be the number of possible tests $\zeta$ (see Theorem 2.5), and let $\tau$ be the size of each test (where we pad tests that are not long enough), so that each test $\zeta$ can be written as $(\zeta_1, \ldots, \zeta_\tau)$ with $\zeta_i \in [L]$. Let $\mathcal{U}_{\mathsf{nsPCP},t}$ be the Turing machine that checks each test, as in Theorem 2.5.

Fix an $\ell$-SSB hash family

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

(see Construction 3).

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

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

Let $\mathcal{L}$ be the corresponding language. Notice that the size of an instance is $m = \tau \cdot \log L + n + \ell \cdot (\ell_{\mathsf{hk}} + \ell_{\mathsf{hash}})$. When $\kappa$ (and thus $\ell, \ell_{\mathsf{hk}}, \ell_{\mathsf{hash}}$) is $\mathrm{polylog}(t)$, then $m = \mathrm{polylog}(t) + n$.

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

**Theorem 5.3.** *For $\mathrm{poly}(n) \le t = t(n) \le \exp(n)$, the algorithms $(\mathsf{Setup}_{\mathcal{L}_{\mathcal{U}}(t)}, \mathcal{P}_{\mathcal{L}_{\mathcal{U}}(t)}, \mathcal{V}_{\mathcal{L}_{\mathcal{U}}(t)})$ defined in Figure 5 satisfy the following properties:*

- **Correctness:** *For every $(M, x) \in \mathcal{L}_{\mathcal{U}}(t)$,*

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

**SNARG for $\mathcal{L}_{\mathcal{U}}(t)$**

Fix a SNARG $(\mathsf{Setup}_{\mathsf{Batch}\mathcal{L}}, \mathcal{P}_{\mathsf{Batch}\mathcal{L}}, \mathcal{V}_{\mathsf{Batch}\mathcal{L}})$ for Batch$\mathcal{L}$, as in Definition 5.2. For $\epsilon > 0$, define $\kappa = (\log t)^{2/\epsilon}$ and let $\lambda = \mathrm{polylog}(t)$ be such that $S(\lambda, m, N) \geq 2^{\ell \cdot \ell_{\mathsf{hash}}}$.

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

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

  It also samples

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

  and outputs $\mathsf{crs} = (\mathsf{hk}_{\ell\text{-SSB}}, \mathsf{crs}_{\mathsf{Batch}\mathcal{L}})$.

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

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

  It then computes $\sigma_{\mathsf{Batch}\mathcal{L}} \leftarrow \mathcal{P}_{\mathsf{Batch}\mathcal{L}}(\mathsf{crs}_{\mathsf{Batch}\mathcal{L}}, Y, W)$, where

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

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

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

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

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

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

- **Soundness:** *Assuming that there exists $\epsilon > 0$ such that the underlying* SSB *hash family is $2^{\kappa^\epsilon}$-hiding, and assuming the existence of $\lambda = \mathrm{polylog}(t)$ such that the* BatchNP *SNARG is $S$-sound for $S(\lambda, m, N) \geq 2^{\ell \cdot \ell_{\mathsf{hash}}}$, where $\ell = \kappa \cdot \mathrm{polylog}(t)$ is the size of the queries produced by $\mathcal{Q}_{\mathsf{nsPCP},t}(1^\kappa)$ and $\ell_{\mathsf{hash}}$ is the size of the output of the* SSB *hash family, for any $\mathrm{poly}(t)$-size $\mathcal{P}^*$ and $(M, x) \notin \mathcal{L}_{\mathcal{U}}(t)$,*

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

- **Runtimes:** *The 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 of Theorem 5.3.* Correctness is straightforward. For the complexity analysis, note that the prover first hashes the PCP, which takes time $\mathrm{poly}(t)$, and then emulates the prover from the Batch$\mathcal{L}$ SNARG, which definitionally runs in time $\mathrm{poly}(\lambda, m, N) = \mathrm{poly}(t)$ (Definition 5.2). The

proof string $\sigma$ satisfies $|\sigma| = |\mathsf{rt}| + |\sigma_{\mathsf{Batch}\mathcal{L}}| = \mathrm{poly}(\kappa) + \mathrm{poly}(\lambda, m, \log N) = \mathrm{poly}(n, \log t)$. The verifier simply emulates $\mathcal{V}_{\mathsf{Batch}\mathcal{L}}$, which runs in time $\mathrm{poly}(\lambda, m, \log N) = \mathrm{poly}(n, \log t)$. We now focus on proving soundness.

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

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

This is equal to

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

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

Let $\delta' = \delta - \mathrm{negl}$. Then, the above is equivalent to:

$$\begin{aligned}
\delta'(t) &\leq \Pr\left[\mathcal{V}_{\mathcal{L}_{\mathcal{U}}(t)}(\mathsf{crs}, (M,x), \sigma)) = 1 \,\middle|\, \begin{array}{r} \mathsf{crs} \leftarrow \mathsf{Setup}_{\mathcal{L}_{\mathcal{U}}(t)}(1^\kappa, 1^\lambda) \\ \sigma = (\mathsf{rt}, \sigma_{\mathsf{Batch}\mathcal{L}}) \leftarrow \mathcal{P}^*(\mathsf{crs}, (M,x)) \text{ s.t.} \\ \mathcal{V}_{\mathsf{nsPCP},t}((M,x), Q, \mathsf{Invert}_{\ell\text{-}\mathsf{SSB}}(\mathsf{td}_{\ell\text{-}\mathsf{SSB}}, \mathsf{rt})) = 0 \end{array}\right] \\
&= \Pr\left[\mathcal{V}_{\mathsf{Batch}\mathcal{L}}(\mathsf{crs}_{\mathsf{Batch}\mathcal{L}}, \langle Y \rangle, \sigma_{\mathsf{Batch}\mathcal{L}})) = 1 \,\middle|\, \begin{array}{r} \mathsf{crs} = (\mathsf{hk}_{\ell\text{-}\mathsf{SSB}}, \mathsf{crs}_{\mathsf{Batch}\mathcal{L}}) \leftarrow \mathsf{Setup}_{\mathcal{L}_{\mathcal{U}}(t)}(1^\kappa, 1^\lambda) \\ \sigma = (\mathsf{rt}, \sigma_{\mathsf{Batch}\mathcal{L}}) \leftarrow \mathcal{P}^*(\mathsf{crs}, (M,x)) \text{ s.t.} \\ Y \notin \mathsf{Batch}\mathcal{L} \end{array}\right],
\end{aligned}$$

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

We will use $\mathcal{P}^*$ to break the $S$-security of the Batch$\mathcal{L}$ SNARG as follows. By an averaging argument, there is some $\mathsf{hk}^*_{\ell\text{-}\mathsf{SSB}}$ for which $\mathcal{P}^*(\mathsf{crs}, (M,x))$ outputs $(\mathsf{rt}, \sigma_{\mathsf{Batch}\mathcal{L}})$ with $Y \notin \mathsf{Batch}\mathcal{L}$ and $\mathcal{V}_{\mathsf{Batch}\mathcal{L}}(\mathsf{crs}_{\mathsf{Batch}\mathcal{L}}\langle Y \rangle, \sigma_{\mathsf{Batch}\mathcal{L}}) = 1$ with probability $\geq \delta'(t)$ conditioned on $\mathsf{crs} = (\mathsf{hk}^*_{\ell\text{-}\mathsf{SSB}}, \mathsf{crs}_{\mathsf{Batch}\mathcal{L}})$. Furthermore, there is some $\mathsf{rt}^*$ for which, with probability $\geq \frac{\delta'(t)}{2^{\ell \cdot \ell_{\mathsf{hash}}}}$, this occurs and the rt output by

$\mathcal{P}^*$ is equal to $\mathsf{rt}^*$. In particular, $Y^* \notin \mathsf{Batch}\mathcal{L}$ where $Y^*$ is defined by $\langle Y^* \rangle = ((M, x), \mathsf{hk}^*_{\ell\text{-}\mathsf{SSB}}, \mathsf{rt}^*) \notin \mathsf{Batch}\mathcal{L}$.

$$\Pr\left[\begin{array}{r|l} \mathcal{V}_{\mathsf{Batch}\mathcal{L}}(\mathsf{crs}_{\mathsf{Batch}\mathcal{L}}, \langle Y^* \rangle, \sigma_{\mathsf{Batch}\mathcal{L}})) = 1 & \mathsf{crs}_{\mathsf{Batch}\mathcal{L}} \leftarrow \mathsf{Setup}_{\mathsf{Batch}\mathcal{L}}(1^\lambda, 1^m, N) \\ \wedge\ \mathsf{rt} = \mathsf{rt}^* & \mathsf{crs} := (\mathsf{hk}^*_{\ell\text{-}\mathsf{SSB}}, \mathsf{crs}_{\mathsf{Batch}\mathcal{L}}) \\ & \sigma = (\mathsf{rt}, \sigma_{\mathsf{Batch}\mathcal{L}}) \leftarrow \mathcal{P}^*(\mathsf{crs}, (M, x)) \end{array}\right] \geq \frac{\delta'(t)}{2^{\ell \cdot \ell_{\mathsf{hash}}}} \geq \delta''(S(\lambda, m, N)),$$

where $\delta''$ is a non-negligible function (such $\delta''$ exists since we assumed $S(\lambda, m, N) \geq 2^{\ell \cdot \ell_{\mathsf{hash}}}$). Then, a cheating prover for the Batch$\mathcal{L}$ SNARG, given $Y^* \notin \mathsf{Batch}\mathcal{L}$, can run $\mathcal{P}^*$ on inputs $\mathsf{crs} = (\mathsf{hk}^*_{\ell\text{-}\mathsf{SSB}}, \mathsf{crs}_{\mathsf{Batch}\mathcal{L}})$ and $(M, x)$, where $\mathsf{crs}_{\mathsf{Batch}\mathcal{L}} \leftarrow \mathsf{Setup}_{\mathsf{Batch}\mathcal{L}}(1^\kappa, 1^m, N)$, to get $(\mathsf{rt}, \sigma_{\mathsf{Batch}\mathcal{L}})$. When the Merkle root $\mathsf{rt}$ that $\mathcal{P}^*$ output is equal to $\mathsf{rt}^*$, he outputs $\sigma_{\mathsf{Batch}\mathcal{L}}$, which fools $\mathcal{V}_{\mathsf{Batch}\mathcal{L}}$ with probability non-negligible in $S(\lambda, m, n)$. Furthermore, this Batch$\mathcal{L}$ SNARG cheating prover runs in time $\mathrm{poly}(t) \leq \mathrm{poly}(S(\lambda, m, n))$, as $S(\lambda, m, n) \geq 2^{\ell \cdot \ell_{\mathsf{hash}}} = 2^{\mathrm{polylog}(t)} = \omega(t)$. This contradicts the $S$-security of the Batch$\mathcal{L}$ SNARG. □

## References


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

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

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

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

[BHK17] Zvika Brakerski, Justin Holmgren, and Yael Tauman Kalai. Non-interactive delegation and batch NP verification from standard computational assumptions. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 474–482, 2017. 5, 6, 9, 13, 14

[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. 5, 9, 14

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

[BMW99] Ingrid Biehl, Bernd Meyer, and Susanne Wetzel. Ensuring the integrity of agent-based computations by short proofs. In *Proceedings of the Second International Workshop on Mobile Agents*, MA '98, pages 183–194, London, UK, UK, 1999. Springer-Verlag. 3, 11

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

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

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

[CGKS95] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, USA, 23-25 October 1995*, pages 41–50, 1995. 12

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

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

[DHRW16] Yevgeniy Dodis, Shai Halevi, Ron D. Rothblum, and Daniel Wichs. Spooky encryption and its applications. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part III*, pages 93–122, 2016. 5, 6

[DLN+04] Cynthia Dwork, Michael Langberg, Moni Naor, Kobbi Nissim, and Omer Reingold. Succinct proofs for NP and spooky interactions. Unpublished manuscript, 2004. http://www.cs.bgu.ac.il/~kobbi/papers/spooky_sub_crypto.pdf. 5, 6

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

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

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

[GK16] Shafi Goldwasser and Yael Tauman Kalai. Cryptographic assumptions: A position paper. In *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part I*, pages 505–522, 2016. 7, 8

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

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

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

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

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

[HW15] Pavel Hubácek and Daniel Wichs. On the communication complexity of secure function evaluation with long output. In Tim Roughgarden, editor, *Proceedings of the*

*2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015, Rehovot, Israel, January 11-13, 2015*, pages 163–172. ACM, 2015. 2, 4, 5, 10, 11

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

[JKKZ21] Ruta Jawale, Yael Tauman Kalai, Dakshita Khurana, and Rachel Zhang. Snargs for bounded depth computations and PPAD hardness from sub-exponential LWE. 2021. 1, 21

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *STOC*, pages 723–732, 1992. 2, 11

[KO97] Eyal Kushilevitz and Rafail Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *FOCS*, pages 364–373, 1997. 12, 13

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

[KRR13] Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. Delegation for bounded space. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 565–574, 2013. 5

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

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

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

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

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

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