Boosting Batch Arguments and RAM Delegation

Boosting Batch Arguments and RAM Delegation*

Yael Kalai
Microsoft Research and MIT
USA
yael@microsoft.com

Alex Lombardi
Simons Institute and UC Berkeley
USA
alexlombardi@alum.mit.edu

Vinod Vaikuntanathan
MIT
USA
vinodv@mit.edu

Daniel Wichs
Northeastern University and NTT Research
USA
wichs@ccs.neu.edu

ABSTRACT

We show how to *generically* improve the succinctness of non-interactive publicly verifiable batch argument (BARG) systems. In particular, we show (under a mild additional assumption) how to convert a BARG that generates proofs of length $\mathrm{poly}(m) \cdot k^{1-\epsilon}$, where m is the length of a single instance and k is the number of instances being batched, into one that generates proofs of length $\mathrm{poly}(m, \log k)$, which is the gold standard for succinctness of BARGs. By prior work, such BARGs imply the existence of SNARGs for deterministic time T computation with succinctness $\mathrm{poly}(\log T)$.

Our result reduces the long-standing challenge of building publicly-verifiable delegation schemes to a much easier problem: building a batch argument system that *beats the trivial construction*. It also immediately implies new constructions of BARGs and SNARGs with polylogarithmic succinctness based on either bilinear maps or a combination of the DDH and QR assumptions.

Along the way, we prove an *equivalence* between BARGs and a new notion of SNARGs for (deterministic) RAM computations that we call "*flexible* RAM SNARG*s with partial input soundness*." This is the first demonstration that SNARGs for deterministic computation (of any kind) imply BARGs. Our RAM SNARG notion is of independent interest and has already been used in a recent work on constructing rate-1 BARGs (Devadas et. al. FOCS 2022).

CCS CONCEPTS

• **Theory of computation → Cryptographic protocols**; **Interactive proof systems**; **Cryptographic primitives**.

KEYWORDS

Delegation of Computation, Batch Arguments, RAM Delegation

ACM Reference Format:
Yael Kalai, Alex Lombardi, Vinod Vaikuntanathan, and Daniel Wichs. 2023. Boosting Batch Arguments and RAM Delegation. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing (STOC '23), June 20–23, 2023, Orlando, FL, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3564246.3585200

*The full version of this paper is available at [11].

STOC '23, June 20–23, 2023, Orlando, FL, USA

ACM ISBN 978-1-4503-9913-5/23/06.
https://doi.org/10.1145/3564246.3585200

1 INTRODUCTION

Efficient verification of computation is one of the most fundamental problems in theoretical computer science. Recently, with the increasing popularity of blockchain technologies and cloud services, efficient verification schemes are increasingly deployed in practice. This reality motivates the study of *succinct non-interactive arguments* (SNARGs) [17], which are short, easy to verify, and computationally sound proofs that a statement x belongs to a potentially complex language $\mathcal{L}$. We would ideally like to construct SNARGs, given a (short, efficient-to-generate) common reference string, for any language $\mathcal{L}$ decidable in non-deterministic time $T(|x|)$, where the SNARG has proof size $\mathrm{poly}(\lambda, \log T)$ and verification time $\mathrm{poly}(\lambda, \log T) + \tilde{O}(|x|)$ given security parameter λ. In the random oracle model, such SNARGs were already constructed in a seminal work of Micali [17]; however, constructing SNARGs in the "plain model" under falsifiable and preferably standard cryptographic assumptions remains a grand challenge, and will require overcoming some serious barriers [7].

In this work, we study two different forms of SNARGs for *restricted* computations: SNARGs for *deterministic* time-T computations and SNARGs for *batch*-NP computations (BARGs).

SNARG*s for deterministic time-T computations.* There are a number of recent (We use SNARGs to refer to publicly verifiable SNARGs, which can be verified given the crs alone. We mention that there is a line of work, starting with [14, 15], that constructed various privately verifiable SNARGs under standard assumptions. Since our focus is on publicly verifiable SNARGs, we do not elaborate on these works here.) constructions of this form of SNARG based on falsifiable and standard assumptions:

- Kalai, Paneth and Yang [13] constructed SNARGs with succinctness $\mathrm{poly}(\lambda, T^{\epsilon})$ for any constant $\epsilon > 0$ under a falsifiable assumption on groups with bilinear maps.
- Jawale, Kalai, Khurana and Zhang [10] constructed SNARGs for any size S and depth D (log-space uniform) computation with succinctness $D \cdot \mathrm{poly}(\lambda, \log S)$ under LWE.
- Very recently, Choudhuri, Jain and Jin [4] constructed $\mathrm{poly}(\lambda, \log T)$-succinct SNARGs from LWE.

SNARG*s for batch* NP *computations (*BARG*s).* In a BARG scheme, the prover wants to prove k NP statements $x_1, \ldots, x_k$ (given witnesses $w_1, \ldots, w_k$) with communication complexity (and verification time) significantly smaller than $k \cdot m$, where each witness w_i

is at most m bits long. The key parameter of interest here is k, the batch size, and we want protocols with a sub-linear (and ideally, poly-logarithmic) dependence on k.

There have also been a number of recent constructions of BARGs from standard assumptions:

- Choudhuri, Jain and Jin [3] constructed a BARG with succinctness $\mathsf{poly}(\lambda, m) \cdot \sqrt{k}$ under a combination of the Quadratic Residuosity (QR) and Decisional Diffie-Hellman DDH assumptions (or QR and LWE, but this is subsumed below).
- The work of [4] constructed a BARG with succinctness $\mathsf{poly}(\lambda, m, \log k)$ under LWE. Indeed, their BARG was the key building block in their construction of a SNARG for deterministic polynomial-time computations.
- Building upon [3], Hulett, Jawale, Khurana and Srinivasan [9] constructed a BARG with succinctness $\mathsf{poly}(\lambda, m) \cdot k^{\epsilon}$ for any constant $\epsilon > 0$, under QR and DDH.
- More recently, Waters and Wu [19] constructed a BARG with succinctness $\mathsf{poly}(\lambda, m) \cdot k^{\epsilon}$ under the DLIN assumption (They can rely on the "k-LIN assumption" for arbitrary constant $k \geq 1$, which we refrain from writing due to notation collision with the batch size k. We will sometimes refer to this as the $O(1)$-LIN assumption.) on bilinear maps.

BARG*s imply* SNARG*s for* P. The recent constructions of BARGs and SNARGs for P are quite closely tied together: two recent works [4, 16] showed that a BARG for batch NP with L-parameterized (These works only considered the case $L(k, \lambda) = \mathsf{poly}(\lambda, \log(k))$, but their results readily extend to any L.) succinctness $\mathsf{poly}(m) \cdot L(k, \lambda)$ *implies* a SNARG for any time-T computation with succinctness $\mathsf{poly}(\lambda) \cdot L(T, \lambda)$ assuming the existence of somewhere extractable succinct commitments with local opening. (We show in the full version that such commitments can be constructed from any rate-1 string OT, which in turn can be constructed based on any of the following assumptions: LWE, DDH, $O(1)$-LIN, QR, or DCR [6].)

The [4, 16] transformation requires that the underlying BARG satisfy one key property regarding *verification time* that we will also assume throughout this work. (In [4], a BARG with this efficiency guarantee was referred to as a BARG for index languages. In this work, we actually only assume the existence of an object that is somewhat weaker than an index BARG; see the full version for more details.) Namely, the BARG must be compatible with *succinct implicit inputs*: if a time $\ell = \ell(n)$ Turing machine generates x_i given i as input (for all i), then verifying the BARG for $(x_1, \ldots, x_k)$ can be done in time polynomial in ℓ and (as before) sublinear or better in $k \cdot m$. Note that this means the verifier does not necessarily have to *read* the k statements (Since a verifier (in general) must read its entire input in order to decide whether to accept a proof, a naive BARG for k NP statements would require the verifier to run for at least $k \cdot n$ time. This is unsatisfactory if the goal is to have efficiency sublinear in k.) (separately), but only their implicit description. Throughout this introduction, when we refer to a BARG, we assume it has this verifier efficiency guarantee.

These two works changed the focus of the community from constructing SNARGs to constructing BARGs. Indeed, the recent BARG constructions of [4, 9, 19] all imply constructions of similarly efficient SNARGs for P.

1.1 This Work

Both primitives discussed above – SNARGs for P and BARGs for NP – have a "gold standard" for succinctness and verifier efficiency:

- SNARGs for time T (deterministic) computation with communication complexity $\mathsf{poly}(\lambda, \log T)$.
- BARGs with communication complexity $\mathsf{poly}(\lambda, m, \log k)$.
- For both BARGs and SNARGs, verification can require quasi-linear additional time to process explicit inputs but should otherwise match the communication complexity bounds.

However, of all of the recent constructions discussed so far, only [10] (for bounded-depth deterministic computation) and [4] actually match this efficiency. The others [3, 9, 13, 19] achieve *sublinear* (and sometimes sub-polynomial) but not polylogarithmic dependence on T and/or k. In this work, our main question centers on achieving optimal succinctness for BARGs and SNARGs:

Question 1.1. *When is it possible to build* BARG*s and* SNARG*s with* polylogarithmic *(w.r.t. k or T) succinctness and verifier efficiency?*

A second question we ask is whether there is any (partial) converse to the [4, 16] result that BARGs imply SNARGs for P:

Question 1.2. *Does some kind of* SNARG *for deterministic computation imply* BARG*s for* NP*?*

A positive answer would establish a (loose) *equivalence* between these two kinds of argument systems.

Our Results. We answer Question 1.1 by giving a generic procedure for *boosting* the efficiency of BARGs. Specifically, we show how to convert any BARG with succinctness $\mathsf{poly}(m) \cdot k/p(\lambda)$, for some (sufficiently large) polynomial $p(\lambda)$, into one with succinctness $\mathsf{poly}(\lambda, m, \log k)$.

Theorem 1.3 (Informal, see Theorem 3.1). *There is a polynomial p such that if there exists*

- *A* BARG *for* NP *with succinctness* $\mathsf{poly}(m) \cdot k/p(\lambda)$ *for all sufficiently large $k \geq \mathsf{poly}(\lambda)$ (and efficient verifier), and*
- *A rate-1 (2-message) string OT, which can be constructed based on* LWE, DDH, $O(1)$-LIN, QR, *or* DCR [6].

then there exists a BARG *for* NP *with succinctness* $\mathsf{poly}(\lambda, m, \log k)$ *(and efficient verifier).*

We briefly give some remarks on Theorem 1.3:

- A BARG with succinctness matching the hypothesis of Theorem 1.3 follows from the existence of a BARG with succinctness $\mathsf{poly}(\lambda, m) \cdot k^{1-\delta}$ for any constant $\delta > 0$.
  - If we make a subexponential security assumption, it is even possible to start with any BARG with succinctness $\mathsf{poly}(\lambda, m) \cdot \frac{k}{(\log k)^{\omega(1)}}$, by setting the security parameter $\lambda = \mathsf{poly}\log(k\lambda')$. However, the resulting fully succinct BARG will only be secure against adversaries that run in time quasi-polynomial in $m\lambda'$, which is meaningful but not ideal.
- Theorem 1.3 can start with BARGs with a *long* ($\mathsf{poly}(k, m, \lambda)$) common reference string, as long as the verifier efficiency remains $\mathsf{poly}(m) \cdot k/p(\lambda)$ (taking as input only a designated part of the crs of appropriate size). This is because a simple

transformation can be used to first reduce the size of the crs while preserving the above sublinear succinctness bound: to prove k statements, pick a small constant $\epsilon > 0$ and execute $k^{1-\epsilon}$ copies of the initial BARG with batch size k^{ϵ} (re-using a single crs).

- The rate-1 string OT is used (solely) to construct a somewhere extractable hash (SEH) family with local opening . This matches (For simplicity of the write-up, our SEH definition differs slightly from what was used in [4]. We make use of a form of *deterministic* succinct commitment schemes, while [4] allows randomization. However, such schemes can always be derandomized with a (public seed) PRF, so the primitives are equivalent.) what is required in the generic transformations of [4, 16]. We state Theorem 1.3 using string OT simply to highlight the variety of instantiations of the building block, many of which were not previously known .
- We emphasize that our results (and proofs) are *entirely in the setting of non-interactive arguments*. Unlike prior work such as [2–4, 9, 10], we do *not* make explicit use of interactive proofs or the Fiat-Shamir transform, but instead generically convert a weakly succinct (non-interactive) BARG into a strongly succinct BARG.

Theorem 1.3, combined with the works of [4, 16], reduces the problem of constructing ideal SNARGs for time-T computations to constructing *any non-trivial* BARG, one that is slightly more succinct that simply sending all the NP witnesses in the clear.

As corollaries to Theorem 1.3, we obtain multiple new constructions of BARGs for NP and SNARGs for P:

- Together with the result of [19], Theorem 1.3 gives a BARG for NP with proof size $\mathsf{poly}(\lambda, m, \log k)$ and a SNARG for time-T computations of size $\mathsf{poly}(\lambda, \log T)$ (as opposed to size $\mathsf{poly}(\lambda, m, k^{\epsilon})$ and $\mathsf{poly}(\lambda, T^{\epsilon})$) from DLIN (or $O(1)$-LIN) on bilinear maps. Moreover, this BARG can be obtained from the "base" scheme of [19] without their "bootstrapping" step.
- Together with the result of [3], Theorem 1.3 gives BARGs and SNARGs with the above efficiency from QR and DDH, as opposed to having a $\sqrt{k}$ dependence in [3] or a k^{ϵ} (or T^{ϵ}) dependence in [9].

Perhaps more importantly, we believe Theorem 1.3 is an important foundation that will lead to new constructions of BARGs and SNARGs, since it reduces this goal to a significantly easier problem.

In order to prove Theorem 1.3, we also obtain an answer to Question 1.2 by considering the setting of RAM delegation [1, 12]. We define (and construct) a new notion of RAM SNARG, which we call a *flexible* RAM SNARG with *partial input soundness*. We then prove:

Theorem 1.4 (informal, see Theorem 3.3). *Assuming the existence of rate-1 string OT,* BARG*s for* NP *are existentially* equivalent *to flexible* RAM SNARG*s with partial input soundness.*

We prove both directions of this equivalence (assuming rate-1 string OT):

(1) BARGs for NP imply flexible RAM SNARGs in an efficiency-preserving manner. This is a strengthening of the [4, 16] result, which only constructs a weaker form of RAM delegation from BARGs.
(2) Flexible RAM SNARGs, *even with barely non-trivial succinctness*, imply BARGs with succinctness $\mathsf{poly}(\lambda, m, \log k)$.

Sequentially combining these two transformations yields Theorem 1.3. Combining them the opposite order also implies that the succinctness of flexible RAM SNARGs can be boosted.

In addition to facilitating Theorem 1.3, we believe that our notion of flexible RAM SNARGs is of independent interest; indeed, it has already been used to obtain a simplified rate-1 BARG in [5].

2 OUR TECHNIQUES

Somewhere extractable BARG*s (*seBARG*s).* Before diving into our techniques, we first simplify our problem by replacing BARGs with a slightly stronger primitive *without loss of generality*. Namely, we consider *somewhere extractable* BARGs, hereafter referred to as seBARGs. A BARG is defined to be somewhere extractable if for some (hidden) choice of i, given an appropriate trapdoor td for the crs, it is possible to *extract* a witness w_i for x_i given a valid BARG proof. This is essentially an argument of knowledge property for BARGs.

Conveniently, assuming the existence of somewhere extractable hash functions with local opening, BARGs can easily be modified to be somewhere extractable with the standard "commit-and-prove" approach (for example, this was used implicitly in [4, 16]). From now on we work directly with seBARGs instead of BARGs, since seBARGs are an easier-to-manipulate primitive.

Organization. We now give an overview of our proofs of both directions of Theorem 1.4; these together also imply Theorem 1.3. We begin by recalling RAM delegation and give intuition for why it should be useful for constructing BARGs. We then discuss our new notion of flexible RAM SNARGs with partial input soundness and sketch our main proofs.

RAM *delegation.* A RAM SNARG, originally defined in [12], is similar to a SNARG for deterministic computations but tailored to the RAM computational model. Concretely, we consider the simplified setting of read-only RAM computation. A (read-only) RAM algorithm is given query-access to a large input x (often referred to as its "memory") and returns some output y. Queries to memory are considered unit-cost operations.

In a RAM SNARG, the prover wants to convince the verifier that $M(x) = y$ for some RAM machine M, input x, and output y. In addition to wanting verification that is efficient compared to the *runtime* of $M(x)$, it is also desired that verification is sublinear (preferably polylogarithmic) in the length of the input x. However, the verifier must have some "handle" on x in order for verification to be possible; to do so, the verifier is given a *digest* $\mathsf{d} = \mathsf{Digest}(\mathsf{crs}, x)$, which can roughly be thought of as a Merkle tree commitment to x.

Given this syntax, we arrive at an important question: what does it mean for a RAM SNARG to be *sound*? Observe that the map $x \mapsto \mathsf{Digest}(\mathsf{crs}, x)$ is many-to-one, so the input x is not information-theoretically defined from the point of view of the verifier. Thus, $\mathsf{Digest}(\mathsf{crs}, \cdot)$ is always required to be collision-resistant, capturing the intuition that the prover should be committed to some particular input x when it sends d.

In prior work [1, 4, 12, 13], soundness was formulated in two different ways:

- In [1, 12], a RAM SNARG is defined to be sound if it is computationally hard to prove two *contradictory* statements; namely, to prove that both $M(x) = 0$ and $M(x) = 1$ with the same machine M and digest d (note that a specific input x does not necessarily exist in this security notion; the prover may not actually know one).
- In [4, 13], a *weaker* security property was used: a RAM SNARG was defined to be sound if it is computationally hard to *simultaneously* (1) make the verifier accept with machine M, digest d, and output y and (2) produce an input x such that $M(x) \neq y$ and $\mathsf{Digest}(\mathsf{crs}, x) = \mathsf{d}$. Note that the [1, 12] security definition implies this one.

Previous constructions of RAM SNARGs from standard assumptions [4, 13] (and those that follow by combining [4] with [3, 9, 19]) were only shown to satisfy the weaker of the above two definitions. We emphasize that this soundness definition is quite weak: soundness is guaranteed only against an adversary that "knows" the entire memory x corresponding to a digest d. In this work, we revisit the notion of RAM SNARGs and provide a new, stronger definition of soundness that overcomes this weakness and facilitates Theorems 1.3 and 1.4.

How to use SNARG*s for* RAM *to build* BARG*s.* Before getting to our new definition, let us sketch why SNARGs for RAM are useful for constructing BARGs; the connection is surprisingly simple in hindsight. At a high level, the idea is for the prover P to treat its k NP witnesses $w_1, \ldots, w_k$ as the *memory* of a RAM machine. Thus, the prover will compute $\mathsf{d} = \mathsf{Digest}(\mathsf{crs}, w_1, \ldots, w_k)$ (where crs is associated with some SNARG for RAM) and send d to the verifier.

Now, the most naive approach would be for the prover to send a SNARG that $w_1, \ldots, w_k$ are all valid witnesses for $x_1, \ldots, x_k$, but it is completely unclear how to argue soundness of the resulting BARG relying on any soundness property of the SNARG. (We remark that in the privately verifiable setting, the batch argument system of [1] is somewhat similar to the "naive" construction above, but they do *not* rely on a RAM SNARG. Instead, they rely on what later became known as a *quasi-argument* for NP [13], which is a much more powerful building block.) The problem is that fundamentally, there is no way to guarantee that an adversarial prover P^* who makes the verifier accept (with digest d) actually *knows* the contents of a memory $(w_1, \ldots, w_k)$ that corresponds to d.

Instead, we will have the prover produce k *different* SNARGs $\pi_1, \ldots, \pi_k$ on memory $(w_1, \ldots, w_k)$ with respect to k different RAM computations. Specifically, we define the ith RAM computation M_i to consider only the ith "chunk" of memory and verify that w_i is a valid witness for x_i. An initial candidate BARG can then be the digest d along with $\pi_1, \ldots, \pi_k$; the verifier simply checks each π_i separately (with respect to d).

Although we have not argued soundness, this is already a non-trivial candidate BARG! As long as each π_i is significantly shorter than m (the length of w_i), the communication complexity of this protocol will be significantly shorter than the trivial bound of $k \cdot m$. This establishes an intuitive connection between RAM SNARGs and BARGs.

Challenges in Arguing Soundness. Despite having a simple candidate BARG with non-trivial efficiency, soundness of this candidate is not obvious and in fact does not seem to follow from previous security definitions for RAM SNARGs. In fact, the problem seems similar to the "naive" case: soundness of a RAM SNARG is only guaranteed against adversaries that "know" the entire memory, which is $w_1, \ldots, w_k$, but there is no way to argue that an adversary P^* must know such a long string (since the BARG itself is short).

However, there is a key difference from before: each RAM computation M_i only operates on a *small fraction* of its memory, namely, w_i (ignoring all w_j for $j \neq i$). Moreover, if Digest is *somewhere extractable* [8] on m locations (henceforth called a m-SEH), it *is* possible to argue that an adversary P^* (at least inside a security reduction) knows the fraction of RAM memory that is *relevant* to any particular M_i. This opens up the possibility for the following kind of security proof:

- Suppose that P^* is a convincing prover for the BARG, and let i be an index such that the statement x_i is false.
- Switch to a hybrid experiment in which the crs is *statistically binding* (and extractable) on w_i. In this hybrid, it is possible to produce both a valid BARG proof (where x_i is false (In this overview, we assume for simplicity that the statements $x_1, \ldots, x_k$ are fixed in advance. The situation is more subtle if the x_i are chosen adaptively, but non-trivial security properties can be argued.)) and obtain the unique w_i consistent with d.
- Argue that the proof π_i produced by P^* contradicts the soundness of the RAM SNARG.

Unfortunately, previous RAM SNARG security definitions [1, 4, 12, 13] are not compatible with this security reduction. As a result, we next revisit and revise the foundations of RAM delegation.

Flexible SNARG*s for* RAM. There are two significant issues with previous notions of RAM SNARGs if we want to use them. First of all, we want a RAM SNARG with the property that the Digest algorithm is *somewhere extractable* on m locations. This begs the question: do such RAM SNARGs exist? More generally, one can ask: which additional properties can the Digest algorithm of a RAM SNARG potentially have?

We address these questions by defining *flexible* RAM SNARGs, which are a generic RAM SNARG *template* making use of an arbitrary hash family with local opening. (A hash family with local opening is a deterministic, computationally binding succinct commitment to a long string x along with a procedure for producing a short ($\mathrm{poly}(\lambda)$-size) opening to any bit x_i. This commitment need not hide information about x. In this paper, we relax the standard definition to allow for the hash key and hash output to each have two parts: (potentially long) sender components $(\mathsf{hk}, \mathsf{v})$ and (short) receiver components $(\mathsf{vk}, \mathsf{rt})$ (which are used for opening verification).) Specifically, a flexible RAM SNARG is a scheme defined relative to a generic hash family, which plays the role of the Digest algorithm. A flexible RAM SNARG has the property that for *any* secure hash family with local opening, the resulting RAM SNARG is sound. In other words, flexible RAM SNARGs imply that *any hash family with local opening* can be used as the Digest algorithm

for a RAM SNARG. This tells us that we can plug in a Digest algorithm that is somewhere extractable, provided that it *also* has local openings.

Partial-Input Soundness. The second major problem with RAM SNARGs is that, as stated earlier, they provide no security guarantees against adversaries who produce a digest d^* *without knowledge* of a *full opening* of d^* to an input x (representing the full contents of a machine's memory). This is problematic in scenarios where a SNARG is used to prove statements about RAM machines that only access a small fraction of their memory.

This motivates defining soundness (or argument of knowledge) properties for RAM delegation schemes in situations where the adversary does *not* know an entire input x. Formulating such security notions can be quite subtle. In this work, for simplicity, we focus on the following setting:

- The Digest hash function is *extractable* on some set of locations S. This means that given an arbitrary d^*, it is possible to extract an assignment x_S so that an opening of d^* to any location $i \in S$ *must* reveal the bit x_i.
- The RAM machine M, when run on any memory consistent with x_S, only reads locations in S. This is equivalent to the assertion that this holds when M is run on the specific input x^* such that $x^*_i = x_i$ for $i \in S$ and $x^*_i = 0$ otherwise.

In this situation, we say that a RAM SNARG satisfies *partial-input soundness* if it is computationally hard to produce a machine M, digest d^*, output y, and proof π such that

- The verifier accepts $(M, \mathsf{d}^*, y, \pi)$, and
- $M(x^*) \neq y$, where x^* is obtained from d^* as above, and $M(x^*)$ only reads locations in S.

We emphasize that this definition does not require that the adversary possesses an opening (A natural alternative soundness definition would simply require that it is computationally hard for P^* to produce accepting (M, y, d, π) and local openings to a substring x_S such that $M(x^*)$ only reads locations in S and $M(x^*) \neq y$. However, for technical reasons, this turns out to be an *insufficient* definition to support our transformations, since we cannot always guarantee (in our soundness reductions) that P^* knows how to open x_S.) of d^* to x_S; nevertheless, the string x_S is well-defined (and efficiently accessible) in the security game.

Armed with this definition, we return to our main results relating BARGs and RAM SNARGs: assuming the existence of rate-1 string OT, the following two claims hold.

Claim 2.1. seBARG*s imply flexible* RAM SNARG*s with partial-input soundness.*

Claim 2.2. *Flexible* RAM SNARG*s with partial-input soundness imply* seBARG*s. This transformation* boosts *succinctness from "non-trivial" to* $\mathsf{poly}(\lambda, m, \log k)$.

So far, we sketched a weak variant of Claim 2.2 that does *not* boost succinctness. We conclude by discussing how to prove Claim 2.2 (in full) and Claim 2.1.

Boosting Succinctness via Recursion. Recall our candidate non-trivial seBARG making use of a (flexible) RAM SNARG:

- The prover sends a digest $\mathsf{d} = \mathsf{Digest}(w_1, \ldots, w_k)$, and
- The prover sends k SNARGs $\pi_1, \ldots, \pi_k$ associated with d, where π_i is a proof that w_i is a valid witness for x_i.

We indeed show that this construction is sound assuming that Digest is somewhere extractable and the RAM SNARG satisfies partial-input soundness. However, this argument system is only *somewhat* succinct: the size of the proof is $|\mathsf{d}| + \sum |\pi_i|$, which grows linearly with k. Can we do better?

The answer is that we can by adapting an insight from [4] to our setting. (One can view [4] as implementing the following strategy: (1) construct a weakly succinct interactive batch argument scheme, (2) extend this particular scheme to a fully succinct interactive batch argument scheme, and (3) compile it into a BARG using the Fiat-Shamir transform [2]. (2) is accomplished via an interactive recursion.

Theorem 1.3 suggests an alternative approach: (1′) build a weakly succinct interactive scheme, (2′) *apply the Fiat-Shamir transform right away* to get a weakly succinct BARG, and (3′) invoke Theorem 1.3 to boost the succinctness generically.) Namely, we observe that the proof string $(\mathsf{d}, \pi_1, \ldots, \pi_k)$ has *reduced* the problem of verifying $w_1, \ldots, w_k$ to the easier problem of verifying $\pi_1, \ldots, \pi_k$. Provided that the time to verify each π_i is at most half the time required to verify each w_i, we can *pair* adjacent proofs (π_{2i-1}, π_{2i}) together and obtain a batch NP verification problem with $k/2$ witnesses of complexity no larger than that of the original w_i. Then, instead of sending these witnesses (the π_i) in the clear, we can have the prover recursively run our protocol: send $\mathsf{Digest}(\pi_1, \ldots, \pi_k)$ and compute proofs $\pi'_1, \ldots, \pi'_{k/2}$ certifying that all pairs (π_{2i-1}, π_{2i}) would be accepted by the RAM SNARG verifier. This recursion can be executed $\log k$ times in total, resulting in a seBARG in which the prover sends $\log k$ digests $\mathsf{d}_0, \ldots, \mathsf{d}_{\log k - 1}$, where d_i is a digest of $k/2^i$ strings $\pi^{(i)}_1, \ldots, \pi^{(i)}_{k/2^i}$ that are RAM SNARG proofs computed with respect to d_{i-1}. At the end of the recursion, there will be a single RAM SNARG proof $\pi^{(\log k)}$ that the verifier can receive and check on its own.

Crucially, we observe that as long as the RAM SNARG is *non-trivially succinct* – meaning that the computational cost of verifying a pair (π_1, π_2) is lower than the cost of verifying a single NP witness w – then the resulting seBARG will have ideal succinctness $\mathsf{poly}(\lambda, m, \log k)$. This is what enables our generic boosting results; see the full version for more details.

Fully Local Hashing. So far, we have sketched how to construct seBARGs given a flexible RAM SNARG with partial-input soundness, when the Digest algorithm for the RAM SNARG is somewhere extractable on m locations. Next, we address a technical issue with this approach.

The problem is that flexible SNARGs for RAM are only as as efficient as the underlying Digest algorithm plugged into them. Specifically, the *verification time* of the SNARG grows with the size of a local opening for Digest. However, a hash family that is somewhere extractable on m locations must necessarily have an output of length $\geq m$ [8], so opening verification would seem to require at least m time (even to read the hash value). This would result in a RAM SNARG whose verification time grows with m, which for our candidate BARG above would be useless: the BARG's size would be larger than $k \cdot m$.

This incompatibility is resolved with the recently introduced notion of a "fully local (somewhere extractable) hash family" [5]. In such a hash family $\mathcal{H}$, a hash evaluation can be divided into two parts: a long (length $\geq m$) component v and a short (length $\mathsf{poly}(\lambda)$) component rt. (Similarly, the hash key can be divided into a long component hk and a short component vk.) The hash family is then required to be *extractable* given v and have *local openings* of rt of size $\mathsf{poly}(\lambda)$ to individual input bits. Finally, consistency between v and rt is enforced; $\mathsf{rt} = \mathcal{H}.\mathsf{Digest}(\mathsf{crs}, \mathsf{v})$ is a fixed function of v.

A fully local SEH hash family $\mathcal{H}$ resolves our technical issue and enables a provable construction of seBARGs from flexible RAM SNARGs. But how is this building block instantiated? [5] constructed such hash families from the LWE assumption. Their construction was complicated and required powerful tools (including rate-1 FHE and seBARGs themselves!), but they were aiming for a *rate*-1 fully local hash family.

In this work (see the full version), we give a simple construction of a (low rate) flSEH family from any SEH family with local opening. That is, we show that SEH families that are binding on a single index (or on m indices with openings that grow linearly with m) can be generically made fully local.

At a high level, our flSEH family is constructed as follows. Suppose that we want to hash inputs $x \in \{0,1\}^n$ in a way that is extractable on m indices $i_1, \ldots, i_m \in [n]$. Since a m-SEH requires openings of size $\geq m$ (as discussed above), we instead *separately* hash x m *different times* using SEH functions $h_1, \ldots, h_m$. Each $h_j \leftarrow \mathcal{H}$ is set up to be extractable on $\mathsf{poly}(\lambda)$ locations and have openings of size $\mathsf{poly}(\lambda)$. The resulting m hash values $\mathsf{v}_1, \ldots, \mathsf{v}_m$ are defined to be the extractable hash output v; v is then *digested* using a hash tree into a root rt of size $\mathsf{poly}(\lambda)$. To open a bit x_i with respect to the root rt, a hash function index j is selected *pseudorandomly* (The notion of pseudorandomness required is that of a *load-balancing* hash function: the n indices $\{1, \ldots, n\}$ should be mapped pseudorandomly into m buckets so that for any m-tuple $(i_1, \ldots, i_m)$ no bucket has more than $\mathsf{poly}(\lambda)$ of those indices in it.) (as a function of i), v_j is opened (w.r.t. rt), and then x_i is opened (w.r.t. v_j). This strategy enables us to make the overall hash family extractable on index i by making the hash function h_j extractable on i, and thus allows for m-location extractability with $\mathsf{poly}(\lambda)$-size openings. See the full version for details.

Having resolved the local opening subtlety, this completes our overview of Claim 2.2.

Constructing our RAM SNARG*s from* seBARG*s.* Finally, we turn to Claim 2.1: constructing flexible RAM SNARGs with partial input soundness from seBARGs. This construction additionally uses a SEH family with local opening and closely follows the transformations of [4, 16]. To give a succinct proof that $M(x) = y$ with respect to a digest d (computed with respect to an *arbitrary* Digest algorithm with local opening), compute a somewhere extractable hash $h = \mathsf{SEH.Hash}(\mathsf{st}_1, \ldots, \mathsf{st}_T)$, where $\mathsf{st}_1, \ldots, \mathsf{st}_T$ denotes the sequence of memory configurations for the execution of $M(x)$. Then, generate and send a seBARG that, roughly speaking, $\mathsf{st}_i \rightarrow \mathsf{st}_{i+1}$ for all i. More formally, the seBARG is executed on a batch NP statement whose witnesses are openings to pairs $(\mathsf{st}_i, \mathsf{st}_{i+1})$ along with openings of d to the bit x_j that st_i asks to read; the NP relation checks that the correct bit is read and that the state transformation $\mathsf{st}_i \rightarrow \mathsf{st}_{i+1}$ is executed correctly. Since this batch of NP statements has a succinct representation (given by d along with h and the hash keys), the resulting RAM SNARG will be as succinct (up to $\mathsf{poly}(\lambda, |\mathsf{st}|)$ factors) as the BARG.

The main difference from the [4, 16] setting is that we wish to prove *partial-input soundness*, which states that a (malicious) prover cannot produce a digest d and accepting SNARG proof (M, y, π) such that $M(x^*) \neq y$, where x^* is constructed by extracting a substring x_S from d and setting all other x_i to 0. This follows from a hybrid argument combining the (extractable) binding property of Digest with the (extractable) soundness property of the seBARG. Essentially, it is possible to argue sequentially that for every time-step t, if the seBARG crs is set to be extractable on the tth NP statement, then the extracted state st_t must match the state of $M(x^*)$ at time t. We refer the reader to the full version for more details. We also remark that as a side result, we prove that our construction satisfies the original [1, 12] definition of soundness, which is incomparable to partial-input soundness.

This completes our sketch of Claim 2.1. Combining Claim 2.1 and Claim 2.2 appropriately, we obtain Theorem 1.3 and Theorem 1.4.

2.1 Relation to [5]

A recent work of Devadas, Goyal, Kalai, and Vaikuntanathan [5] (concurrently with a work of Paneth and Pass [18]) constructs seBARGs that have *rate 1* with respect to the size of an NP witness. The notion of "rate-1 fully local hash" was introduced in an initial version of [5] for their construction; we then used a relaxation of their notion (a fully local hash family that has *low rate*) in this work. Subsequently, an updated version of [5] gives a significantly simplified construction of rate-1 seBARGs that leverages our notion of flexible RAM SNARGs adapted to their rate-1 setting.

3 FORMAL RESULTS

In this section, we formally state our results (Theorems 1.3 and 1.4) and show how they follow from the results proved in the full version of this paper ([11] Sections 3 to 7).

Theorem 3.1. *Assume the existence of rate-*1 *String OT with verifiable correctness ([11] Definitions 4.1 and 4.2), or more generally a* SEH *family with succinct local opening ([11] Definition 3.3).*

Then, there exists an explicit polynomial $p(\lambda)$ such that the following holds.

Let $L(k, \lambda)$ denote any function such that $L(k, \lambda) \leq k/p(\lambda)$ for all sufficiently large $k \geq \mathsf{poly}(\lambda)$. Assuming the existence of a $L(k, \lambda)$-succinct index BARG *for* BatchCSAT*, there exists a* $\mathsf{poly}(\lambda, \log k)$*-succinct index* BARG *for* BatchCSAT*. Moreover, there exists a* $\mathsf{poly}(\lambda, \log T)$*-succinct* SNARG *for* P *(and for* RAM *computation).*

Remark 3.1. As discussed in the introduction,

- Index BARGs with efficiency $\mathsf{poly}(m, \lambda)k^{1-\delta}$ for any constant $\delta > 0$ suffice for the $L(k, \lambda)$ hypothesis, and thus imply fully succinct BARGs (assuming a SEH).
- Index BARGs with efficiency $\mathsf{poly}(m, \lambda)\frac{k}{(\log k)^{\omega(1)}}$ with sub-exponential security also suffice by setting $\lambda = \mathsf{poly}\log(k \cdot \lambda')$ for a new security parameter λ'. The resulting fully succinct BARG will only be secure against adversaries that run

in time quasi-polynomial in $m \cdot \lambda'$, as the proof of Theorem 3.1 calls the weak BARG with batch size $\mathsf{poly}(m)$. This is a significant drawback but still a meaningful BARG.

As discussed in the full version, the use of index BARGs in these two instantiations could be replaced with the use of (non-index) BARGs for BatchCSAT. However, this remains a stronger assumption than the existence BARGs for L^k for some NP-complete language L.

Proof Sketch of Theorem 3.1. [11] Lemma 4.5 tells us that rate-1 String OT satisfying verifiable correctness implies an SEH family with succinct local opening. In turn, [11] Theorem 5.2 implies that an SEH family with succinct local opening implies the existence of a fISEH family ([11] Definition 5.1).

We proceed to prove Theorem 3.1 by a composition of several transformations.

- By [11] Lemma 3.9, $L(k,\lambda)$-succinct index BARGs for BatchCSAT (along with a SEH family with local opening) imply $L^{(2)}(k,\lambda) = L(k,\lambda) \cdot \mathsf{poly}(\lambda)$-succinct index seBARGs for BatchCSAT.
- By [11] Theorem 6.3, $L^{(2)}(k,\lambda)$-succinct index BARGs for BatchCSAT (along with a SEH family with local opening) imply $L^{(3)}(T,\lambda) = L(T,\lambda) \cdot \mathsf{poly}(\lambda, \log T)$-succinct flexible SNARGs for RAM with partial-input soundness.
- By [11] Theorem 7.1, $L^{(3)}(T,\lambda)$-succinct flexible SNARGs for RAM with partial-input soundness imply polylog-succinct index seBARGs provided that $L^{(3)}(T,\lambda) \leq T/\lambda$ for sufficiently large $T \geq T_0(\lambda)$.
- Finally, by [4, 16] we already know that polylog-succinct index seBARGs imply SNARGs for P and for RAM. By [11] Theorem 6.3, they in fact even imply flexible RAM SNARGs with partial-input soundness.

Since each of the transformations can be implemented in a way that incurs a *fixed* $\mathsf{poly}(\lambda)$ overhead, the theorem follows. □

Corollary 3.2. *There exist* $\mathsf{poly}(\lambda, \log k)$*-succinct index* BARG*s for* BatchCSAT *and* $\mathsf{poly}(\lambda, \log T)$*-succinct* SNARG*s for* P *under either*

(1) *The* $O(1)$-LIN *assumption on a pair of cryptographic groups with efficient bilinear map, or*
(2) *A combination of the sub-exponential* DDH *assumption and the* QR *assumption.*

Proof. By [11] Appendix A, we know that under any of the DDH, QR, and $O(1)$-LIN assumptions, there exists a rate-1 string OT scheme to fulfill the hypothesis of Theorem 3.1.

Moreover, [19] constructed an index-BARG scheme for BatchCSAT with sublinear succinctness under $O(1)$-LIN on bilinear maps. They first construct a scheme with (polylogarithmic online communication and) a large crs of size $\mathsf{poly}(k, m, \lambda)$; instead of reducing the crs size by using Section 5 of [19], we can simply execute $k^{1-\delta}$ copies of the scheme with batch size k^{δ} (re-using the same short crs) to immediately obtain sublinear overall succinctness (choosing small enough $\delta < 1/2$).

Additionally, [3] constructed (Index BARGs were not defined in [3], but it is easily seen that their construction satisfies the required efficiency property. [3] Corollary 1 and Corollary 2 establish $(|C| + k)\mathsf{poly}(\lambda)$ efficiency for C-index languages; by combining groups of $\sqrt{k}$ statements together, we obtain sublinear succinctness. Similarly, the notion of semi-adaptive soundness was not defined in [3], but their unmodified construction satisfies it.) an index-BARG scheme for BatchCSAT with sublinear succinctness under sub-exponential DDH and QR

Given these building blocks, the claimed results follow by Theorem 3.1. □

Theorem 3.3. *Assume the existence of rate-*1 *String OT with verifiable correctness ([11] Definitions 4.1 and 4.2), or more generally a* SEH *family with succinct local opening ([11] Definition 3.3).*

Then, $\mathsf{poly}(\lambda, \log k)$*-succinct index* BARG*s for* BatchCSAT *exist if and only if* $\mathsf{poly}(\lambda, \log T, \log N)$*-succinct flexible* RAM SNARG*s with partial-input soundness ([11] definition 6.2) exist.*

Proof. [11] Lemma 4.5 tells us that the String OT building block implies an SEH family with succinct local opening. In turn, [11] Theorem 5.2 implies that an SEH family with succinct local opening implies the existence of a fISEH family.

The equivalence can then be established as follows:

- By [11] Lemma 3.9, succinct index BARGs for BatchCSAT (along with a SEH family with local opening) imply succinct index seBARGs for BatchCSAT.
- By [11] Theorem 6.3, succinct seBARGs for BatchCSAT (along with a SEH family with local opening) imply flexible RAM SNARGs with partial-input soundness.
- By [11] Theorem 7.1, flexible RAM SNARGs with partial-input soundness (along with a fISEH family) imply succinct BARGs for BatchCSAT. □

ACKNOWLEDGEMENTS

We thank the anonymous STOC reviewers for their helpful feedback.

This research was conducted in part while AL was at MIT, where he was supported by a Charles M. Vest Grand Challenges Fellowship, an NDSEG Fellowship, and grants of VV. VV is supported in part by DARPA under Agreement No.
HR00112020023, a grant from the MIT-IBM Watson AI, a grant from Analog Devices and a Microsoft Trustworthy AI grant. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Government or DARPA.

REFERENCES

[1] Zvika Brakerski, Justin Holmgren, and Yael Tauman Kalai. 2017. 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*, Hamed Hatami, Pierre McKenzie, and Valerie King (Eds.). ACM, 474–482. https://doi.org/10.1145/3055399.3055497 3, 4, 6

[2] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. 2019. Fiat-Shamir: from practice to theory. In *51st ACM STOC*, Moses Charikar and Edith Cohen (Eds.). ACM Press, 1082–1090. https://doi.org/10.1145/3313276.3316380 3, 5

[3] Arka Rai Choudhuri, Abhishek Jain, and Zhengzhong Jin. 2021. Non-interactive Batch Arguments for NP from Standard Assumptions. In *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part IV*. 394–423. 2, 3, 4, 7

[4] Arka Rai Choudhuri, Abhishek Jain, and Zhengzhong Jin. 2021. SNARGs for $\mathcal{P}$ from LWE. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*. IEEE, 68–79. https://doi.org/10.1109/FOCS52979.2021.00016 1, 2, 3, 4, 5, 6, 7

[5] Lalita Devadas, Rishab Goyal, Yael Kalai, and Vinod Vaikuntanathan. 2022. Rate-1 Non-Interactive Arguments for Batch-NP and Applications. In *Proceedings of FOCS 2022*. 3, 6

[6] Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. 2019. Trapdoor Hash Functions and Their Applications. In *CRYPTO 2019, Part III (LNCS, Vol. 11694)*, Alexandra Boldyreva and Daniele Micciancio (Eds.). Springer, Heidelberg, 3–32. https://doi.org/10.1007/978-3-030-26954-8_1 2

[7] Craig Gentry and Daniel Wichs. 2011. Separating succinct non-interactive arguments from all falsifiable assumptions. In *43rd ACM STOC*, Lance Fortnow and Salil P. Vadhan (Eds.). ACM Press, 99–108. https://doi.org/10.1145/1993636.1993651 1

[8] Pavel Hubacek and Daniel Wichs. 2015. On the Communication Complexity of Secure Function Evaluation with Long Output. In *ITCS 2015*, Tim Roughgarden (Ed.). ACM, 163–172. https://doi.org/10.1145/2688073.2688105 4, 5

[9] James Hulett, Ruta Jawale, Dakshita Khurana, and Akshayaram Srinivasan. 2022. SNARGs for P from Sub-exponential DDH and QR. In *EUROCRYPT 2022, Part II (LNCS, Vol. 13276)*, Orr Dunkelman and Stefan Dziembowski (Eds.). Springer, Heidelberg, 520–549. https://doi.org/10.1007/978-3-031-07085-3_18 2, 3, 4

[10] Ruta Jawale, Yael Tauman Kalai, Dakshita Khurana, and Rachel Yun Zhang. 2021. SNARGs for bounded depth computations and PPAD hardness from sub-exponential LWE. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, Samir Khuller and Virginia Vassilevska Williams (Eds.). ACM, 708–721. https://doi.org/10.1145/3406325.3451055 1, 2, 3

[11] Yael Tauman Kalai, Alex Lombardi, Vinod Vaikuntanathan, and Daniel Wichs. 2022. Boosting Batch Arguments and RAM Delegation. Cryptology ePrint Archive, Report 2022/1320. https://eprint.iacr.org/2022/1320. 1, 6, 7

[12] Yael Tauman Kalai and Omer Paneth. 2016. Delegating RAM Computations. In *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*. 91–118. https://doi.org/10.1007/978-3-662-53644-5_4 3, 4, 6

[13] Yael Tauman Kalai, Omer Paneth, and Lisa Yang. 2019. How to delegate computations publicly. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*. 1115–1124. 1, 2, 4

[14] Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. 2013. Delegation for bounded space. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*. 565–574. https://doi.org/10.1145/2488608.2488679 1

[15] Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. 2014. How to delegate computations: the power of no-signaling proofs. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*. 485–494. https://doi.org/10.1145/2591796.2591809 1

[16] Yael Tauman Kalai, Vinod Vaikuntanathan, and Rachel Yun Zhang. 2021. Somewhere Statistical Soundness, Post-Quantum Security, and SNARGs. In *Theory of Cryptography - 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8-11, 2021, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 13042)*, Kobbi Nissim and Brent Waters (Eds.). Springer, 330–368. https://doi.org/10.1007/978-3-030-90459-3_12 2, 3, 6, 7

[17] Silvio Micali. 1994. CS Proofs (Extended Abstracts). In *35th FOCS*. IEEE Computer Society Press, 436–453. https://doi.org/10.1109/SFCS.1994.365746 1

[18] Omer Paneth and Rafael Pass. 2022. Incrementally Verifiable Computation via Rate-1 Batch Arguments. In *63rd FOCS*. IEEE Computer Society Press, 1045–1056. https://doi.org/10.1109/FOCS54457.2022.00102 6

[19] Brent Waters and David J Wu. 2022. Batch Arguments for NP and More from Standard Bilinear Group Assumptions. *Cryptology ePrint Archive* (2022). 2, 3, 4, 7

Received 2022-11-07; accepted 2023-02-06