Succinct Arguments from Multi-Prover Interactive Proofs and their Efficiency Benefits*

Nir Bitansky†
nirbitan@tau.ac.il
TAU

Alessandro Chiesa
alexch@csail.mit.edu
MIT

December 1, 2012

Abstract

Succinct arguments of knowledge are computationally-sound proofs of knowledge for NP where the verifier's running time is independent of the time complexity t of the nondeterministic NP machine M that decides the given language. Existing succinct argument constructions are, typically, based on techniques that combine cryptographic hashing and probabilistically-checkable proofs (PCPs). Yet, even when instantiating these constructions with state-of-the-art PCPs, the prover needs $\Omega(t)$ space in order to run in quasilinear time (i.e., time $t \cdot \mathrm{poly}(k)$), regardless of the space complexity s of the machine M.

We say that a succinct argument is *complexity preserving* if the prover runs in time $t \cdot \mathrm{poly}(k)$ and space $s \cdot \mathrm{poly}(k)$ and the verifier runs in time $|x| \cdot \mathrm{poly}(k)$ when proving and verifying that a t-time s-space random-access machine nondeterministically accepts an input x. Do complexity-preserving succinct arguments exist? To study this question, we investigate the alternative approach of constructing succinct arguments based on multi-prover interactive proofs (MIPs) and stronger cryptographic techniques:

(1) We construct a one-round succinct MIP of knowledge, where each prover runs in time $t \cdot \mathrm{polylog}(t)$ and space $s \cdot \mathrm{polylog}(t)$ and the verifier runs in time $|x| \cdot \mathrm{polylog}(t)$.

(2) We show how to transform any one-round MIP protocol to a succinct four-message argument (with a single prover), while preserving the time and space efficiency of the original MIP protocol; using our MIP protocol, this transformation yields a complexity-preserving four-message succinct argument.

As a main tool for our transformation, we define and construct a *succinct multi-function commitment* that (a) allows the sender to commit to a vector of functions in time and space complexity that are essentially the same as those needed for a single evaluation of the functions, and (b) ensures that the receiver's running time is essentially independent of the function. The scheme is based on fully-homomorphic encryption (and no additional assumptions are needed for our succinct argument).

(3) In addition, we revisit the problem of *non-interactive* succinct arguments of knowledge (SNARKs), where known impossibilities prevent solutions based on black-box reductions to standard assumptions. We formulate a natural (but non-standard) variant of homomorphic encryption having a *homomorphism-extraction property*. We show that this primitive essentially allows to squash our interactive protocol, while again preserving time and space efficiency, thereby obtaining a complexity-preserving SNARK. We further show that this variant is, in fact, implied by the existence of (complexity-preserving) SNARKs.

*An extended abstract of this paper appears in the proceedings of CRYPTO '12. This is the full version.

†Supported by the Check Point Institute for Information Security, a Marie Curie grant PIRG03-GA-2008-230640, an ISF grant 0603805843, and the Fulbright program.

Contents

1 Introduction

Interactive proofs & succinctness. Interactive proofs [GMR89] are central to modern cryptography and complexity theory. One extensively-studied aspect of interactive proofs is their expressibility; this study culminated with the result that $\mathsf{IP} = \mathsf{PSPACE}$ [Sha92]. Another aspect, the focus of this work, is that proofs for NP-statements can potentially be verified much faster than by directly checking an NP witness.

Unfortunately, in interactive proofs with statistical soundness, non-trivial savings in verification time are unlikely [BHZ87, GH98, GVW02, Wee05]. However, if we settle for proof systems with only *computational* soundness (also known as argument systems [BCC88]), then significant savings can be made.

Indeed, using collision-resistant hashes (CRHs) and probabilistically-checkable proofs (PCPs) [BFLS91], Kilian [Kil92] showed a four-message interactive argument where membership of an instance x in an NP language L can be verified in time bounded by $p(k + |M| + |x| + \log t)$, where t is the time to evaluate the NP verification machine M for L on input x and a valid witness, p is a fixed polynomial independent of L, and k is a security parameter. Following tradition, we call such argument systems *succinct*.

A natural strengthening of computational soundness is (computational) *proof of knowledge*: it requires that, whenever the verifier is convinced by an efficient prover, not only can we conclude that a valid witness for the theorem *exists*, but also that such a witness can be *extracted efficiently* from the prover. Proof of knowledge is a natural property (satisfied by most proof system constructions, including the aforementioned one of Kilian [BG08]) that is very useful in many applications of succinct arguments.

A special case of succinct arguments that has received a lot of attention is the one of succinct *non-interactive* arguments of knowledge (SNARKs). Indeed, SNARKs are known for their powerful applications, including: non-interactive delegation of computation, succinct non-interactive secure computation, extractable cryptographic primitives [BCCT12a], and constructions of proof-carrying data [CT10, BCCT12b].

1.1 Existing Succinct Arguments And Their Efficiency

Kilian's four-message succinct argument of knowledge works as follows: the prover first uses a Merkle hash tree to bind itself to a polynomial-size PCP oracle for the statement to be proven, and then answers the PCP verifier's queries while demonstrating consistency with the previous Merkle tree commitment.

Most SNARK constructions [Mic00, DCL08, BCCT12a, DFH12, GLR11] are based on techniques for "squashing" Kilian's protocol into a non-interactive one, and hence are also based on the "commit to a polynomial-size PCP oracle and then reveal" paradigm; informally, we shall refer to a protocol following this paradigm as a *Kilian-type protocol*.

The current state of affairs is unsatisfying in two respects. On the one hand, from an "understanding perspective" it is unsatisfying that the only way we know how to build succinct arguments, at least from standard assumptions, is through Kilian-type protocols. On the other hand, from an efficiency perspective, Kilian-type protocols suffer from inefficiencies that one might in principle hope to avoid — let us explain.

In state-of-the-art PCPs [BSCGT12, BSCGT13], the time complexity of the prover and verifier are essentially optimal: for a random-access machine M, input x, and time bound t, (i) the PCP prover generates a PCP oracle for the statement "there is a witness w for which $M(x, w)$ accepts within t steps" in time $(|M| + |x| + t) \cdot \mathrm{polylog}(t)$ and (ii) the PCP verifier runs in time $(|M| + |x|) \cdot \mathrm{polylog}(t)$. However, the quasilinear running time of the PCP prover is achieved via FFT-like methods, which unfortunately demand $\Omega(t)$ space. (And not using FFTs would make the prover's running time quadratic in t.)

In a Kilian-type protocol, the prover has to generate the entire PCP oracle in order to commit to it. Thus, even when using state-of-the art PCPs in a Kilian-type protocol, to run in time $t \cdot \mathrm{poly}(k)$ (rather than $\Omega(t^2)$),

the prover requires $\Omega(t)$ space, even if the computation of M on input (x, w) requires space s with $s \ll t$.

While in succinct arguments the focus is usually on minimizing the verifier's complexity, in most applications the prover is also of bounded resources and thus minimizing its complexity is also crucial; after all, the verifier may himself be paying to use the prover's resources (e.g., when delegating computation to servers rented from a cloud service). Thus, a desirable goal is to ensure that the computational costs of the prover are as close as possible to those of the original computation. A bit more precisely, we would like to have *complexity-preserving* succinct arguments for NP, where the prover runs in time $(|M| + |x| + t) \cdot \mathrm{poly}(k)$ and space $(|M| + |x| + s) \cdot \mathrm{poly}(k)$ and the verifier runs in time $(|M| + |x|) \cdot \mathrm{poly}(k)$, when proving and verifying that a t-time s-space random-access machine M for an NP language nondeterministically accepts an input x.

Q: *Do complexity-preserving succinct arguments exist?*

What we know about complexity-preserving succinct arguments. Subsequently to our work, Bitansky et al. [BCCT12b] constructed SNARKs that are not of the Kilian-type: their construction relies on recursive composition and "bootstrapping" of SNARKs with an expensive offline preprocessing phase [Gro10, Lip12, GGPR12, BCI$^+$13] (which do not invoke the PCP Theorem but only simpler probabilistic-checking techniques [BCI$^+$13]). The SNARKs constructed in [BCCT12b] via this transformation are in fact complexity preserving.

However, the transformation in [BCCT12b] does not extend to produce complexity-preserving succinct arguments in the interactive setting. In the interactive setting, we may hope to prove security under standard cryptographic assumptions [GW11], and complexity-preserving succinct arguments from such assumptions are not known. Further, even in the non-interactive setting, it would be nice to have a more "direct" construction of a complexity-preserving SNARK, without going through the transformation in [BCCT12b].

Finally, we note that, even for the more modest (but still very desirable) goal of succinct arguments for deterministic polynomial-time computations (rather than NP computations), we *also* do not have complexity-preserving solutions.

Necessity of PCPs: an obstruction to preserving complexity? A potential obstruction to obtaining complexity-preserving succinct arguments is that Rothblum and Vadhan [RV09] proved that PCPs are in a certain sense inherent to succinct argument constructions: any succinct argument (even if interactive) can be transformed into a PCP, as long as its security is established via a black-box reduction to standard cryptographic assumptions. Because at present we only know how to leverage PCPs via Kilian-type protocols, one might interpret the result of Rothblum and Vadhan as saying that the aforementioned inefficiencies of Kilian-type protocols cannot be avoided. The transformation of Rothblum and Vadhan, however, incurs significant overhead in the general case. Thus, it is *still* possible for there to exist a succinct argument construction that is more efficient than a Kilian-type one; this holds *even* if such a construction induces a corresponding PCP. Looking ahead, one of the results of this paper is that this is indeed the case.

2 Summary Of Our Results

At high-level, we show how PCPs in succinct arguments can be replaced by *multi-prover interactive proofs* (MIPs) [BOGKW88], by using appropriate cryptographic tools, to obtain *new and complexity-preserving* succinct argument constructions. Specifically (and as summarized in Figure 1), our results are as follows.

First, we show that, by using existing proximity testing techniques:

Theorem 1: "There is a one-round succinct MIP of knowledge where each MIP prover runs in time $(|M| + |x| + t) \cdot \text{polylog}(t)$ and space $(|M| + |x| + s) \cdot \text{polylog}(t)$ and the MIP verifier runs in time $(|M| + |x|) \cdot \text{polylog}(t)$, when proving and verifying that a t-time s-space random-access machine M nondeterministically accepts an input x with $|x| \leq t$."

With the goal of using our MIP protocol from Theorem 1, we then proceed to the task of "implementing" the MIP model (i.e., construct a succinct argument with a single prover) in a way that *preserves* the efficiency properties of a given MIP protocol. We show that:

Theorem 3: "Assuming the existence of fully-homomorphic encryption, there exists a complexity-preserving four-message succinct argument of knowledge."

The main tool in the above theorem is a *succinct multi-function commitment*, which enables a sender to commit to a vector of functions in time and space complexity that are essentially the same as those needed for a single evaluation of the functions, and where the receiver's running time is essentially independent of the function.

Theorem 2: "Assuming the existence of fully-homomorphic encryption, there is a succinct multi-function commitment."

In addition, we explore methods to construct SNARKs based on MIPs. Here, known impossibilities rule out solutions based on black-box reductions to falsifiable assumptions [GW11]. We formulate and study a natural (but non-falsifiable) variant of homomorphic encryption that we call *homomorphism-extractable encryption* (HEE), and suggest a candidate construction for this primitive. We show that:

Theorem 4: "Assuming the existence of fully-homomorphic encryption that is homomorphism-extractable, there exists a complexity-preserving SNARK. Furthermore, such encryption schemes are implied by the existence of complexity-preserving SNARKs."

The aforementioned variant of homomorphic encryption also yields a generic transformation from public-coin interactive *arguments* (and not only proofs as in [KR09]) to corresponding non-interactive arguments.

In the following section, we discuss and explain in somewhat more detail our results; pointers to technical sections will be provided along the way.

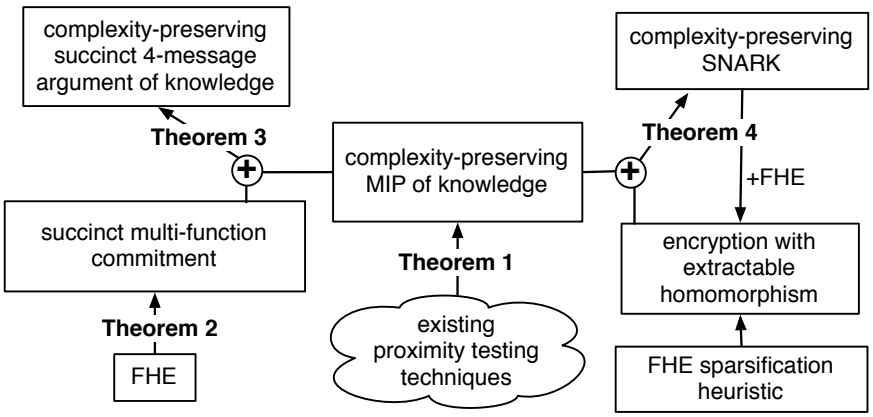

Figure 1: Diagram of our results. For a quick summary see Section 2; for a longer summary see Section 3. Details are contained in the technical sections.

3 Overview Of Our Results

The main idea: functions instead of strings. A seemingly wasteful feature of Kilian-type protocols is that the prover needs to write down an entire PCP oracle Π in order to commit to it; yet, eventually, the verifier is only going to query few locations of Π.

But suppose we had a function f_Π that, on input a location i, outputs Π_i and is relatively cheap to compute. Can we enable the prover to (succinctly) commit to f_Π "as a function" rather than (as in a Kilian-type protocol) asking him to evaluate f_Π everywhere on its domain and then (succinctly) commit to f_Π's evaluation table as a string? In other words, can we develop cryptographic tools that let us implement a *function PCP* model rather than merely the (traditional) *string PCP* model? If we could do so and also find an f_Π that is cheap enough to compute, then we might be able to avoid the inefficiencies of Kilian-type protocols.

Instead of working in a "function PCP" model, we choose to work with an existing model that already has functions as part of its definition: *(one-round) multi-prover interactive proofs* (MIPs). In the MIP model, there are multiple non-communicating provers, and each prover i evaluates his own function f_i on the i-th query q_i of the verifier. (The fact that the functions f_i need not be equal means that the MIP model is slightly easier to implement than the function PCP model; ultimately, though, our techniques will enable us to implement both.)

The idea of succinct function commitments takes inspiration from the work of Ishai et al. [IKO07], who studied the question of whether succinct arguments can be constructed using "lighter" tools than polynomial-size PCPs at the expense of stronger cryptographic assumptions; indeed, Ishai et al. constructed succinct function commitments for linear functions and then used these to obtain "semi-succinct" arguments based on the (exponentially-long) Hadamard code.

We now discuss in more detail each of our results, following the above motivation.

3.1 A Complexity-Preserving 1-Round Succinct MIP Of Knowledge

In the model of *multi-prover* interactive proofs (MIPs), originally introduced by Ben-Or et al. [BOGKW88], the verifier conducts a simultaneous interactive protocol with several provers that, crucially, are assumed to be unable to communicate during the protocol.

We revisit MIPs and show that, despite the fact that PCPs can be used to construct MIPs and vice versa [TS96], our present ability to exhibit efficient constructions for the two is vastly different. Indeed, as explained above, while the traditional PCP model is about strings (i.e., PCP oracles), the MIP model is about *functions* (i.e., the MIP provers); this difference will enable us to exhibit a complexity-preserving MIP construction, while we do not know of any PCP construction with comparable efficiency. Let us explain.

Complexity preservation for PCPs and MIPs. Analogously to the definition of a complexity-preserving succinct argument, we make the following informal definitions:

- A PCP system is *complexity preserving* if, to check that a t-time s-space random-access machine M non-deterministically accepts an input x with $|x| \leq t$, the PCP prover runs in time $(|M|+t)\cdot\mathrm{polylog}(t)$ and space $(|M| + |x| + s) \cdot \mathrm{polylog}(t)$ and the PCP verifier runs in time $(|M| + |x|) \cdot \mathrm{polylog}(t)$.
- An MIP system is *complexity preserving* if, to check that a t-time s-space random-access machine M non-deterministically accepts an input x with $|x| \leq t$, each MIP prover runs in time $(|M| + t) \cdot \mathrm{polylog}(t)$ and space $(|M| + |x| + s) \cdot \mathrm{polylog}(t)$ and the MIP verifier runs in time $(|M| + |x|) \cdot \mathrm{polylog}(t)$.

Our first result is that we are able to construct complexity-preserving MIPs:

Theorem 1. *There is a complexity-preserving one-round MIP of knowledge: to check that a t-time s-space random-access machine M non-deterministically accepts an input x with $|x| \leq t$, each MIP prover runs in time $(|M| + |x| + t) \cdot \mathrm{polylog}(t)$ and space $(|M| + |x| + s) \cdot \mathrm{polylog}(t)$ and the MIP verifier runs in time $(|M| + |x|) \cdot \mathrm{polylog}(t)$. Moreover, each MIP prover can be computed "gate-by-gate" as a circuit by an evaluator algorithm in time $(|M| + |x| + t) \cdot \mathrm{polylog}(t)$ and space $(|M| + |x| + s) \cdot \mathrm{polylog}(t)$.*

Remark 3.1. We do not know how to construct complexity-preserving PCPs (as current state-of-the-art PCPs [BSCGT12, BSCGT13] have essentially-optimal time, but not space, complexity); we find it an intriguing open problem to understand whether these can be constructed or ruled out (see Section 4.)

Our MIP construction ensures that the verifier's message only depends on the time bound t and also ensures a proof of knowledge. Both properties play an important role in the other results in this paper. Another important property is that each MIP prover can be evaluated very efficiently as a circuit.

While Theorem 1 may be surprising (due to the close connection between PCPs and MIPs [TS96]), it has a natural high-level explanation of how it circumvents a seemingly inherent inefficiency of PCPs. Namely, unlike in a PCP, where a prover typically has to evaluate a polynomial at $\Omega(t)$ points, each honest prover in an MIP is typically asked to evaluate a polynomial only at a *single* point, and thus naive evaluation of the polynomial suffices; in particular, no MIP prover has to run FFT-like algorithms that require large space. We next briefly discuss our construction.

Construction outline. We follow a standard paradigm of probabilistic checking by proceeding in two steps:

1. We identify a convenient succinct *algebraic* constraint satisfaction problem sACSP (involving properties of polynomials) and then build on a low-degree proximity test to construct a probabilistic verifier (in our case, relying on the help of multiple non-communicating provers) for this problem.

2. We then show how to reduce computations on random-access machines to sACSP.

We need to ensure that both steps satisfy our stringent efficiency requirements.

More concretely, we proceed as follows.

Step 1. We let sACSP be a modification of a problem described in [Har04, Proposition 5.4.1] and then build a very efficient MIP for sACSP. (Cf. Remark 3.2.) The main ingredients of our construction are the low-degree test of Raz and Safra [RS97] for the Reed–Muller code (i.e., multivariate low-degree polynomials), a technique of Ben-Sasson and Sudan [BSS08, Lemma 4.5] for transforming a low-degree test for the Reed–Muller code to a low-degree test for the *Vanishing* Reed–Muller code (namely, the subcode consisting of those polynomials vanishing on a certain subset of the field), and a simple consistency check.

The technical challenge when constructing an MIP instead of a PCP is that provers are functions and not strings, so that querying a given function at multiple locations is problematic. In our construction, we show how to "distribute" the various functions across as few provers as possible while at the same time ensuring that we do not have to query any given function more than once. (Actually, we will have to query a function more than once only for the consistency check, and to do so additional care will be needed.)

Additionally, we need to ensure that each MIP prover in our construction runs in the appropriate time and space complexity. While at a high level doing so seems plausible (since each MIP prover will essentially be asked to evaluate a certain polynomial at only a single point), from a technical perspective this is not so simple to achieve for each prover since one must ensure that the polynomial itself can be generated "on-the-fly" without using too much time or space (as the polynomial itself is too large too store).

Step 2. We need a reduction from random-access machine computations to SACSP that preserves the time and space complexity of a random-access machine and is succinct (i.e., does not "unroll" the machine's computation); note that succinctness of the reduction is crucial for otherwise the verifier will not have time to invoke it.[1] We construct such a reduction in two substeps. First, we invoke a reduction of Ben-Sasson et at. [BSCGT13] from random-access machine computations to SUCCINCT CIRCUIT SAT (SCSAT); their reduction ensures that a prover can compute a satisfying assignment to the circuit in suitable time and space. Second, we modify the ideas in [Har04, Proposition 5.4.1] to obtain a reduction from SCSAT to SACSP (i.e., we show an arithmetization of SCSAT that preserves succinctness) that again suitably preserves time and space complexities.

For details see Section 5.3 and Section 6.

Remark 3.2. The problem SACSP (which modifies the one of [Har04, Proposition 5.4.1]) that we use here is *not* the same as the succinct algebraic constraint satisfaction problem considered in [BSCGT12] and [BSCGT13]; the former is about multivariate polynomials while the latter is about univariate ones.

Further efficiency benefits. We believe that our MIP construction from Theorem 1 not only has efficiency benefits in the asymptotic sense but also in the concrete sense: our construction does not hide large constants and appears to us as quite simple and efficient — or at least certainly more so than present state-of-the-art PCPs. Let us explain.

We know how to construct either PCPs with great soundness but proofs that are long [RS97, MR08] (and thus with fast verifiers but slow provers), or PCPs with short proofs but soundness that is not as great [BSS08, BSGH$^+$05, BSCGT12] (and thus with not-as-slow provers but quite slow verifiers — due to the large number of repetitions needed to achieve, say, constant soundness).

Obtaining a PCP that *simultaneously* has short proof length and great soundness (or, more accurately, a fast prover and a fast verifier) is an exciting open problem. Ben-Sasson et al. [BSCGT12, BSCGT13], who gave PCPs for random-access machine computations with essentially-optimal time complexity and also studied and optimized their concrete efficiency, show recent progress towards this goal, but their constructions are still not as simple and efficient as one could hope for.

In contrast, with MIPs, there is no such thing as proof length, because the provers are *functions*. In other words, provers do not have to write down long proofs, but only answer specific questions of the verifier, which amounts to just a few evaluations of certain polynomials. Thus, we can design a simple and efficient MIP protocol by using the proximity testing tools at the basis of PCPs with long proofs and relatively high soundness, because we will not be paying for proof length. For example, we can leverage the high soundness of the Subspace vs. Point Test of Raz and Safra [RS97] (and its detailed analysis of Moshkovitz and Raz [MR08]) without worrying about the field size or proximity proof length — and this is indeed what we do.

3.2 Complexity-Preserving Succinct Arguments From MIPs?

Our complexity-preserving MIP construction suggests the possibility of obtaining succinct argument constructions that are complexity preserving. Thus, the question now is:

> Can we "implement" the MIP model with a *single prover* using cryptographic means,
> in a way that preserves its efficiency benefits?

(In particular, we cannot simply deduce a PCP system [TS96] and rely on previous Kilian-type protocols, because doing so would not preserve complexity!) We show two results in this direction, respectively

[1] See [BSCGT13] for more about how reductions affect the efficiency of succinct arguments.

described in Section 3.3 and Section 3.4. Before we do so, however, let us briefly review previous work on MIPs in the context of arguments.

The question of implementing the MIP model for the purpose of constructing succinct arguments was asked by Dwork et al. [DLN$^+$04] with the motivation being non-interactivity, rather than efficiency as in our case. Specifically, after pointing out the unsoundness of the proposed SNARK protocol of Aiello et al. [ABOR00], Dwork et al. suggested that an approach to constructing SNARKs would be to implement the MIP model by encrypting the query for each MIP prover using independent PIR instances and send all the encrypted queries to the same prover. They pointed out that for this approach to work, the PIR should be immune to a specific class of attacks, called "spooky interactions". They were not able to prove any PIR scheme to have this property, nor exhibit an example of a PIR allowing such attacks. By now we know that:

- PIR schemes that prevent spooky interactions cannot be constructed via black-box reductions to falsifiable assumptions [GW11] (though such PIR schemes are implied by the existence of SNARKs); and
- *standard* PIR schemes are in fact sufficient provided that the MIP protocol is sound against spooky interactions as well (i.e., is secure against no-signaling, rather that only non-communicating, provers) [KRR12]; unfortunately, such MIP protocols can at most support computations that lie inside P.

Along a different direction, the work of [IKO07], mentioned above, constructed a compiler from *linear* MIP protocols to four-message interactive arguments by leveraging *commitments with linear decommitment* (which can be obtained, e.g., from additively-homomorphic encryption schemes); by plugging into their compiler a linear MIP based on the Hadamard code, Ishai et al. obtained an argument where the prover-to-verifier communication is small, but the verifier running time is not succinct.

3.3 A Complexity-Preserving Four-Message Succinct Argument Of Knowledge

We now describe how we implement the MIP model using standard cryptographic assumptions and obtain a complexity-preserving four-message succinct argument of knowledge.

A simple but wasteful solution. Given a one-message succinct MIP (of knowledge) protocol and a collision-resistant hash family, it is easy to construct a four-message succinct argument (of knowledge): in the first message the verifier sends to the prover a seed for a collision-resistant hash; then the prover commits (separately and in parallel) to the evaluation table of each MIP prover by sending a Merkle commitment for each; then the verifier sends to the prover the desired message for each MIP prover, and finally the prover replies with an answer to each message, accompanied by an authentication path relative to the appropriate root. Committing to the evaluation table of each MIP prover before any messages are sent by the verifier ensures that a malicious prover cannot "correlate" its answers depending on the messages sent by the verifier in the following message.

However, the above approach *does not preserve* the efficiency properties of the MIP protocol. For example, suppose that the first honest prover P_1 of the protocol is a function $f\colon A \to B$ that requires time t to evaluate at a single point. Then, the above approach would require the prover to evaluate f at every point of A, which could in principle require as much time as $|A| \cdot t$. While the MIP protocol could have started out as very efficient, now having to "think" of each prover as a table of values, instead of as a function, could make the resulting construction *much less* efficient.[2]

[2]For example our MIP construction from Theorem 1 would not preserve complexity (and also be very slow!) if we were to evaluate each prover function everywhere on its domain, even if each prover's domain is "only" of polynomial size. Also, for

Even if there are algorithms for evaluating f everywhere on A that are faster than the naive "point-by-point" evaluation, these faster algorithms may come with hefty space requirements compared to the space requirement of a single evaluation of f. (As discussed in Section 1.1, this is for example the case when f is a polynomial and we seek to evaluate f faster, at many points, via the use of FFT techniques.)

Thus, the approach of using Merkle trees to implement the MIP protocol does not seem to take us any further, in terms of efficiency, than previous PCP-based protocols.

Ideally, we would want a way to implement the MIP protocol that *preserves* its efficiency: namely, the resulting succinct argument prover and verifier should have time and space complexities that are the same as in the corresponding MIP protocol, up to $\mathrm{poly}(k)$ factors, where k is the security parameter.

Our complexity-preserving solution. As briefly explained above, the main idea towards a complexity-preserving solution is to "think" of the MIP provers as functions, rather than write down the entire truth tables of these functions. For this to be possible, we define and show how to obtain a generic primitive called a *succinct multi-function commitment* (SMFC), which is a "complexity-preserving analogue" of a Merkle tree commitment for functions.

Concretely, generalizing the idea of commitments with linear decommitment of Ishai et al. [IKO07] to arbitrary functions, we define an SMFC to be a commitment scheme that works as follows. Given a vector of ℓ functions $\vec{f}\colon A \to B$, where each f_i can be evaluated in time t:[3]

- During a commitment phase, the sender and receiver interact; at the end of this phase, the sender has committed to the vector of functions $\vec{f}$; both parties maintain a private state for the next phase.
- During a decommitment phase, the receiver may request the value of $\vec{f}$ at a vector $\vec{\alpha} \in A^\ell$ by interacting with the receiver. At the end of this phase, the receiver either outputs $\vec{\beta} \in B^\ell$ or $\perp$.

The commitment guarantees that, if both parties are honest, $\vec{\beta} = \vec{f}(\vec{\alpha})$. Moreover, the commitment has the following *computational binding* property: for any efficient malicious sender, after the commitment phase, there is some vector of functions $\vec{f^*}$ such that, for any query $\vec{\alpha}$, the receiver either outputs $\vec{f^*}(\vec{\alpha})$ or outputs $\perp$ (except with negligible probability). We do not make the additional requirement that the commitment is hiding (though our techniques can be enhanced to hide the function being committed to).

The commitment is *succinct* in the sense that the sender runs in time $\ell \cdot \mathrm{poly}(t) \cdot \mathrm{poly}(k)$ and the receiver in time $\ell \cdot (\log|A| + \log|B|) \cdot \mathrm{poly}(k)$ (so that the running time of the receiver does not depend on the time required to evaluate the functions in $\vec{f}$). The commitment is *complexity-preserving* if the sender runs in time $\ell \cdot t \cdot \mathrm{poly}(k)$ and space $\ell \cdot s \cdot \mathrm{poly}(k)$, where s is the space complexity each function in $\vec{f}$.

Succinct arguments from succinct multi-function commitments. Let us briefly describe how to obtain succinct arguments, given succinct multi-function commitments. The protocol essentially consists of two phases: in the first, the prover commits to ℓ functions corresponding to the provers $P_1, \ldots, P_\ell$ given by the one-round MIP protocol. Then, in the second phase, the verifier produces the queries $q_1, \ldots, q_\ell$ to the MIP provers, and asks for the corresponding decommitments. In case all the decommitments are valid, and the MIP verifier is satisfied by the corresponding answers, the verifier accepts. The fact that the prover is committed in advance to each one of the ℓ functions ensures that it cannot correlate the answers depending on the joint vector of queries. This allows us to show that:

functions with superpolynomial-size domain, writing out the function's evaluation table is simply not an option (as it is infeasible); for example, Ishai et al. [IKO07] use linear functions with exponential-size domains and thus cannot rely on Merkle commitments.

[3] We can of course consider functions with different domains and ranges and different evaluation times, but for simplicity here we set them equal.

Claim 1. *Assuming the existence of succinct multi-function commitments, there exists a compiler that transforms a given succinct MIP of knowledge into a corresponding succinct argument of knowledge. Moreover:*

- *The round complexity of the succinct argument is the same as that of the function commitment.*
- *If the function commitment and MIP are complexity-preserving so is the resulting succinct argument.*

We show how succinct function commitments can be constructed using fully-homomorphic encryption:

Theorem 2. *Assuming the existence of fully-homomorphic encryption:*

- *There exists a four-message succinct multi-function commitment for all polynomial-time functions.*
- *There exists a complexity-preserving multi-function commitment for any $\vec{f}$ such that each function in $\vec{f}$ can be computed "gate-by-gate" as a circuit by an evaluator algorithm in time t and space s (and complexity-preservation holds with respect to t and s).*

Remark 3.3. For complexity-preservation to hold in Theorem 2, it is required that the FHE evaluator algorithm operates "locally" in a gate-by-gate fashion. Indeed all known FHE algorithms consist of two "local" evaluator algorithms, one for addition and one for multiplication.[4]

Combining Theorem 1, Claim 1, and Theorem 2, we deduce the existence of complexity-preserving succinct arguments of knowledge from standard assumptions:

Theorem 3. *Assuming the existence of fully-homomorphic encryption there exist four-message complexity-preserving succinct arguments for* NP.

Remark 3.4. More precisely, the complexity-preservation in Claim 1 holds as long the function representing each MIP prover has a sufficiently tight *deterministic* reduction to the representation for which the function commitment is complexity-preserving. In our case, our succinct multi-function commitments are complexity preserving for a suitable circuit representation (cf. Theorem 2) and the provers in our MIP construction have a sufficiently tight reduction to this representation (cf. Theorem 1).

Remark 3.5. Theorem 3 can in fact be strengthened, via a more careful proof, to yield *complexity-preserving universal arguments* [BG08]. It is an interesting open problem to understand whether there exists *public-coin* complexity-preserving universal arguments. (See Section 4.)

We now provide more details regarding our construction of succinct multi-function commitments (given by Theorem 2) and then elaborate on how efficiency is preserved.

Constructing succinct multi-function commitments. Succinct multi-function commitments for functions with polynomial-size domains can of course be constructed generically using succinct arguments for NP together with Merkle hashing. On the other hand, as already discussed, such an approach will not work for functions with superpolynomial-size domains and, moreover, existing succinct arguments for NP are not efficient enough to yield complexity-preserving succinct multi-function commitments even for functions with polynomial-size domains.

We also note that for the purpose of compiling MIPs to succinct arguments, it is sufficient to implement function commitments for the minimal class of functions in which the MIP provers can be implemented.[5]

[4] This requirement can be relaxed; see Remark 5.1.

[5] This observation only concerns completeness; binding is not required to restrict the committed to function to a member of this class.

For example, in our MIP construction, the provers consist of low-degree multivariate polynomials.[6] Our proof of Theorem 2 will be based on fully-homomorphic encryption schemes, and will give a succinct multi-function commitment for *all* polynomial-time functions, even with an exponentially-large domain, and *also a complexity-preserving one when $\vec{f}$ can be easily computed as a circuit*.

The first step towards a multi-function commitment scheme is noting that it is enough to construct a succinct (single-)function commitment. Once this is achieved a commitment to a vector of functions is done by independently committing to each one of the functions. Hence, we focus on the single-function case.

The starting point of our construction is the cut-and-choose delegation scheme of Chung et al. [CKV10], which works as follows:

- given a function f to be delegated, during a setup phase, the verifier generates k independent encryptions $c_1, \ldots, c_k$ of 0 and then homomorphically computes

$$\hat{c}_1 := \mathsf{Eval}(f, c_1), \ldots, \hat{c}_k := \mathsf{Eval}(f, c_k) \ ;$$

- afterwards, during the "online" phase, if the verifier wishes to delegate the computation of f on a given input x, the verifier encrypts k times the input x to obtain $c_{k+1}, \ldots, c_{2k}$, and sends, in random order, $c_1, \ldots, c_{2k}$ to the prover;

- the prover then homomorphically evaluates f on each ciphertext to obtain $\hat{c}'_1, \ldots, \hat{c}'_{2k}$ (in some permuted order) and send them to the verifier; and, finally,

- the verifier will check that $\hat{c}'_1 = \hat{c}_1, \ldots, \hat{c}'_k = \hat{c}_k$ and that $\hat{c}'_{k+1}, \ldots, \hat{c}'_{2k}$ all decrypt to the same value.

Crucially, the verifier's check on the challenges is done "on the ciphertexts"; this enables the security reduction to go through. (And there is an attack if the check is instead performed on the underlying ciphertexts!)

In our setting, we do not have a prover and a weak verifier, but a sender and a weak receiver. In particular, we are not interested in the sender computing a specific function f, but instead we are interested in the sender committing to *some* function. Furthermore, we are not willing to allow the receiver to engage with the sender in an expensive setup phase.

We show that that in a sense we can "delegate" the expensive setup phase of the Chung et al. protocol to obtain an interactive function commitment protocol where the receiver is "fully succinct". More precisely, because the sender is the one deciding the function to compute, we can simply ask him during the first round to evaluate the challenges $\hat{c}_1 := \mathsf{Eval}(f, c_1), \ldots, \hat{c}_k := \mathsf{Eval}(f, c_k)$ himself for some function f of his choosing; of course, we cannot do this "in the clear" (as the sender would see the secret challenges), so we conduct the first round under another layer of fully-homomorphic encryption. In the resulting protocol, the receiver is indeed fully succinct.

While the intuition for the security of our modified protocol is strong, proving its binding property does not appear to be trivial. Our security reduction works as follows:

- We first prove a "weak" binding property for a non-amplified version of the protocol. We show that an adversary that is able, with high probability, to open to two different values of the function for the same query during two independent invocations of the decommitment phase can be used to break semantic security in a certain two-prover game.

[6] It may be tempting here to try and use existing polynomial commitment schemes or polynomial delegation schemes from the literature [KZG10, PST11, BGV11, PRV12, FG12]; however, the aforementioned works require from the receiver a long preprocessing phase, and the delegation schemes require that the receiver knows the polynomial ahead of time. Hence, they are not suitable for our needs. See further discussion in Section 4.

- We then augment the weakly-binding protocol so that we can apply a parallel repetition theorem for interactive arguments (such as [Hai09] or [CL10]), and get a protocol that is strongly binding. This is done by also having the decommitment phase executed under a (second) layer of fully-homomorphic encryption. Unlike typical commitments, where this transformation is rather straightforward, in our case this transformation turns out to be more involved, because the decommitment phase is interactive (and naively making it non-interactive causes an undesired computational overhead).

It may be surprising that, while the security reduction of the Chung et al. [CKV10] protocol is quite straightforward, a simple (and "direct") proof of security for our protocol seems much harder to find. Perhaps the increased difficulty may be attributed to the fact that the Chung et al. [CKV10] protocol only has one round and in this case parallel repetition is, typically, "simpler" [CHS05]; in contrast, in our case proving security of the protocol amounts to proving a special case of parallel repetition for interactive arguments with at least four messages for a protocol that is not of the public coin type (or more generally, one for which sampling random protocol continuations [Hai09] is quite challenging).

Sender complexity in our SMFC. The sender in our SMFC protocol is required to doubly homomorphically evaluate each function in the vector $\vec{f}$. In the general case, we do not know how to homomorphically evaluate a t-time function in less than $t^2 \cdot \mathrm{poly}(k)$ time [BSCGT13]; a fortiori, the same holds for double homomorphic evaluation. Thus, for arbitrary functions, the running time of the sender in our SMFC protocol is only bounded by $\ell \cdot t^2 \cdot \mathrm{poly}(k)$.

If, however, we apply our SMFC protocol to a vector of functions $\vec{f}$ where we do know that every function can be computed by a circuit of size at most t, then we can improve the time bound of the sender to $\ell \cdot t \cdot \mathrm{poly}(k)$. Let us explain why.

The homomorphic evaluation algorithm typically consists of two "local algorithms", homomorphic addition and multiplication over $\mathbb{F}_2$, iteratively applied to the gates of a circuit representation of the function to be evaluated; thus, a circuit C can be homomorphically evaluated in time $|C| \cdot \mathrm{poly}(k)$. In fact, the same is true *also* for *double* homomorphic evaluation: each of the local algorithms can be generically reduced with a quadratic blow up to a corresponding circuit; but both of these algorithms run in time $\mathrm{poly}(k)$ (independently of $|C|$), and thus they both have circuits of size $\mathrm{poly}(k)$; overall, homomorphic evaluation of $|C|$ can be computed by a circuit of size $|C| \cdot \mathrm{poly}(k)$. We conclude that when the functions in $\vec{f}$ have tight reductions to circuits, we can achieve the desired bound of $\ell \cdot t \cdot \mathrm{poly}(k)$ for the running time of the sender.

Furthermore, if we also know that each function in our vector of functions $\vec{f}$ can be evaluated as a circuit in time t and space s (by some "gate-by-gate" evaluator algorithm), then we can also bound the space complexity of the sender by $\ell \cdot s \cdot \mathrm{poly}(k)$. Whenever $s \ll t$ (for example, when the circuits computing the functions have a somewhat succinct representation), this better space bound is very attractive.

See Remark 5.1 for more details.

Thus, overall, in our application of the SMFC protocol in Theorem 3, we use homomorphic encryption in a way that enables us to preserve the efficiency of the MIP protocol we construct in Theorem 1. For details, see Section 7 and Section 8. Finally, unlike the applications in, e.g., [KR09, BCCT12a, GLR11], the FHE scheme in our SMFC construction cannot be substituted by a PIR scheme.

3.4 A Complexity-Preserving SNARK

Having discussed in Section 3.3 how to construct a complexity-preserving interactive succinct argument of knowledge from our Theorem 1, we next consider the natural question of whether we can succeed in the analogous task of constructing a complexity-preserving succinct *non-interactive* argument of knowledge (SNARK) from our Theorem 1.

3.4.1 Known SNARK constructions

Before we discuss our results in this direction, we briefly recall existing SNARK constructions.

In the random-oracle model. Micali [Mic00] showed how to construct publicly-verifiable *one-message* succinct non-interactive arguments, in the random oracle model, by applying the Fiat-Shamir paradigm [FS87] to Kilian's protocol [Kil92]; Valiant [Val08] showed that Micali's protocol is a proof of knowledge.

In the plain model. Micali's protocol is essentially as good as one can hope for in the random oracle model. In the plain model, such "totally non-interactive" succinct arguments (against non-uniform provers) are unlikely to exist for "sufficiently hard" NP-languages (as the impossibility results for statistical soundness can be extended to this case). Yet, known impossibility results leave open the possibility of succinct non-interactive arguments in a slightly more relaxed model, where a generator (run by the verifier or a trusted entity) produces ahead of time a short *reference string* σ for the prover and a short *verification state* τ for the verifier (and both strings are independent of the statements to be proven later). The definition of SNARKs in the plain model refers to this more relaxed setting.

A set of works [BCCT12a, DFH12, GLR11] showed how to construct designated-verifier SNARKs (i.e., τ needs to remain secret) from a non-standard cryptographic primitive called *extractable collision-resistant hashes*. The protocol used in these works revisits a previous protocol proposed by Di Crescenzo and Lipmaa [DCL08] who, in turn, followed up on ideas of Dwork et al. [DLN$^+$04] and Aiello et al. [ABOR00] for "squashing" Kilian's protocol using succinct private information retrieval schemes.

As already discussed in Section 1.1, a separate line of research [Gro10, Lip12, GGPR12, BCI$^+$13] showed how to construct (publicly-verifiable and designated-verifier) SNARKs where the generator is allowed to run as long as the computation to be verified. Bitansky et al. [BCCT12b] showed that this handicap can always be removed via a generic transformation that, in fact, always outputs a complexity-preserving SNARK.

In this paper, we shall aim for a more "direct" construction of complexity-preserving SNARKs, based on our Theorem 1.

Provable limitations. Gentry and Wichs [GW11] showed that no non-interactive succinct argument can be proven to be (adaptively) sound via a black-box reduction to a falsifiable assumption (as defined in [Nao03]), even in the designated-verifier case. Their result suggests that non-standard assumptions, such as knowledge (extractability) assumptions may be inherent for constructing succinct *non-interactive* arguments (even if we were to drop the proof of knowledge requirement). Thus, constructing SNARKs by relying on (reasonable) knowledge assumptions might be justified.

3.4.2 Our Theorem

We adopt a similar strategy to the one in [BCCT12a, DFH12, GLR11]:

1. In Kilian's protocol, the main cryptographic tool is collision-resistant hashes, which are used to commit to the PCP string. By defining the non-standard cryptographic primitive of extractable collision-resistant hashes (ECRH), the works mentioned above show that it is possible to commit and reveal to the PCP string in a single message.

   In our protocol from Section 3.3, the main cryptographic tool is fully-homomorphic encryption, which is used to succinctly commit to a vector of functions. Our goal is to define a (non-falsifiable) cryptographic primitive for revealing values of a vector of functions in a *single* message (as opposed to via an interactive commitment scheme) and show that this is enough to obtain a SNARK from MIPs.

2. Then, [BCCT12a] show that ECRHs are necessary to the construction of SNARKs and exhibit several candidate constructions.

   We will also aim at showing that our primitive is necessary and at presenting candidate constructions.

The extractable primitive. We formulate a natural (though non-falsifiable) primitive of *homomorphism-extractable encryption* (HEE): an encryption scheme where homomorphic operations cannot be performed "obliviously". Specifically, these are (public- or secret-key) encryption schemes that are homomorphic with respect to some given class of functions $\mathcal{F}$. The scheme has a specialized decryption algorithm Dec that takes as input a *source* cipher $\mathsf{c} = \mathsf{Enc}_{\mathsf{pk}}(m)$ and an *evaluated* cipher $\hat{\mathsf{c}}$ (which is allegedly the homomorphic evaluation of some function f on c). If $\hat{\mathsf{c}}$ is indeed such an evaluation, then $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{c}, \hat{\mathsf{c}}) = f(m)$; however, if the adversary "obliviously" computes some function of c, we require that $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{c}, \hat{\mathsf{c}}) = \perp$. More precisely, the guarantee is given in terms of extraction:

$$\begin{gathered}\text{For any efficient } \mathcal{A} \text{ there is an efficient extractor } \mathcal{E}_{\mathcal{A}} \text{ such that:}\\ \text{if } \mathcal{A}, \text{ given } \mathsf{c} = \mathsf{Enc}_{\mathsf{pk}}(m), \text{ outputs } \hat{\mathsf{c}} \text{ such that } \mathsf{Dec}_{\mathsf{sk}}(\mathsf{c}, \hat{\mathsf{c}}) = v \neq \perp,\\ \text{then } \mathcal{E}_{\mathcal{A}}, \text{ given the same input } \mathsf{c}, \text{ outputs a function } f, \text{ such that } f(m) = v.\end{gathered}$$

We stress that the homomorphism-extractability property does *not* provide any guarantee regarding the extracted function f (e.g, it may not be in the family $\mathcal{F}$). In particular, the property does not ensure targeted malleability in the sense of, e.g., [BSW12].

The requirement can then be extended to multiple ciphers (and multiple corresponding functions). Coupled with semantic-security, such a primitive effectively yields a two-message succinct multi-function commitment. Indeed, whenever the adversary manages to transform $\mathsf{c}_1 = \mathsf{Enc}_{\mathsf{pk}_1}(q_1), \ldots, \mathsf{c}_\ell = \mathsf{Enc}_{\mathsf{pk}_\ell}(q_\ell)$ into evaluations $\hat{\mathsf{c}}_1, \ldots, \hat{\mathsf{c}}_\ell$ that are accepted by the decryption algorithm, the extractor will output corresponding functions $f_1, \ldots, f_\ell$ that "explain" the evaluations; moreover, because of semantic security, these functions are independent of the plaintext queries. More precisely, for any two vectors of queries $\vec{q}$ and $\vec{q}'$ and malicious adversary $\mathcal{A}$,

$$\left\{ \vec{f} \;\middle|\; \begin{array}{c}(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^k)\\ \vec{\mathsf{c}} \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\vec{q})\\ \vec{\hat{\mathsf{c}}} \leftarrow \mathcal{A}(\mathsf{c})\\ \vec{f} \leftarrow \mathcal{E}_{\mathcal{A}}(\mathsf{c})\end{array} \right\} \quad \text{and} \quad \left\{ \vec{f}' \;\middle|\; \begin{array}{c}(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^k)\\ \vec{\mathsf{c}}' \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\vec{q}')\\ \vec{\hat{\mathsf{c}}}' \leftarrow \mathcal{A}(\mathsf{c}')\\ \vec{f}' \leftarrow \mathcal{E}_{\mathcal{A}}(\mathsf{c}')\end{array} \right\}$$

are computationally indistinguishable distributions over functions.

SNARKs from HEE. Given a fully-homomorphic encryption that is homomorphism-extractable, we are able to obtain a SNARK construction that is, in a sense, a "squashed" version of the interactive argument described in Section 3.3: the SNARK verifier simply generates MIP queries and sends them encrypted, using an homomorphism-extractable encryption scheme, to the SNARK prover; the prover, in turn, homomorphically evaluates each of the MIP provers on the corresponding encrypted query. As before, provided we start from a complexity-preserving MIP with suitable properties and the FHE scheme is local, we obtain a *complexity-preserving* SNARK. We thus prove the following theorem:

Theorem 4.

- *Assuming the existence of fully-homomorphic encryption that is homomorphism-extractable, there exists an efficiency-preserving compiler that transforms a given succinct one-round MIP of knowledge into a*

corresponding SNARK. In particular, there exists a complexity-preserving SNARK. [7]

- *Furthermore, the existence of local fully-homomorphic encryption and complexity-preserving SNARKs implies the existence of homomorphism-extractable fully-homomorphic encryption.*

The first bullet of Theorem 4 can be seen as a complexity-preserving variant of the PIR-based protocol suggested by Dwork et al. [DLN+04] as a heuristic for squashing Kilian's protocol;[8] indeed, the homomorphism-extractability prevents "spooky attacks" so that the MIP model can be implemented non-interactively. It is interesting to contrast this solution with the one of Kalai et al. [KRR12], who rely on *standard* (sub-exponentially secure) FHE to implement the weaker model of no-signaling (rather than non-communicating) provers.

Furthermore, a corollary to our theorem is an analogue of the [KR09] transformation for all public-coin arguments (and not just proofs): indeed, a public-coin argument immediately implies a simple MIP protocol, where prover P_i gets the random coins $r_1, \ldots, r_i$ used in the first i rounds of the protocol. Applying our HEE-based compiler yields a corresponding one-round argument (naturally, succinctness will only be guaranteed if the original public-coin protocol was succinct.)

For details about Theorem 4, see Section 9 and Section 10.

Is the existence of homomorphism-extractable encryption plausible? Leaving efficiency aside and purely from an existential perspective, homomorphism-extractable encryption (HEE) schemes exist as long as extractable collision-resistant hashes (ECRHs) and FHE schemes exist: [BCCT12a] showed how to construct SNARKs from ECRHs, and we will show that SNARKs and FHE together imply HEE schemes. However, HEE schemes built from SNARKs that do not preserve complexity are not efficient enough for our purposes. We thus must look for "direct" constructions of HEE schemes.

Consider, for example, the fully-homomorphic encryption (FHE) scheme of [BV11], based on LWE. Their FHE scheme likely does *not* have homomorphism-extractability: consider an adversary that, given a ciphertext c of some message m, does not apply the honest evaluation algorithm to some function but instead applies some arbitrary transformation to c to obtain a new ciphertext $\hat{\mathsf{c}}$. With high probability, the new ciphertext $\hat{\mathsf{c}}$ will decrypt to some valid plaintext, but such an adversary may not "know" a corresponding function that can be homomorphically evaluated on the original plaintext c to result in the same ciphertext $\hat{\mathsf{c}}$.

Intuitively, the reason is that the ciphertext space of [BV11] is not "sparse": by merely applying an arbitrary operation in the ciphertext space, an adversary can "jump" from a valid ciphertext to another without being aware of what function he is applying on the underlying plaintext.[9]

Nonetheless, there *is* a natural method to generate sparsity in a given FHE scheme, and this method seems to serve as a heuristic for ensuring that the resulting encryption scheme can be plausibly assumed to have homomorphism-extractability.

The idea is to create sparsity via "amplification": we consider a new encryption scheme, built out of the old one, that, in order to encrypt a given message m, encrypts m under many different independently-generated public keys, and the new decryption algorithm will check that a given vector of ciphertexts decrypts to a vector of messages that are *all equal*. For this amplified encryption scheme, it seems that the

[7] Both Remarks 3.4 and 7.6 also apply for this theorem; that is, to obtain complexity preservation, we require that MIP prover has a sufficiently tight *deterministic* reduction to the an appropriate circuit representation, which is indeed the case for our MIP construction. We also require that the homomorphism-extractable FHE scheme has a "local" evaluation algorithm, which is the case in all known FHE schemes.

[8] We stress, once again, that FHE, cannot be substituted with PIR, because that will not preserve the complexity properties of the MIP protocol.

[9] Sparsity also arises as a necessary condition when studying candidate constructions of ECRHs [BCCT12a].

oblivious adversary described earlier does not work anymore: if the adversary only applies algebraic operations to the ciphertexts, he is likely to obtain ciphertexts that decrypt to different values, that is, an invalid ciphertext of the new scheme.

Homomorphism extractability & plaintext awareness. At high-level, a plaintext aware encryption scheme is such where an adversary cannot create a valid cipher, without actually "knowing" the underlying plaintext. Following previous definitions in the random-oracle model, Bellare and Palacio [BP04] formulated several notions of plaintext awareness in the plain model. Not surprisingly, homomorphism-extractable encryption can be shown to imply some forms of plaintext awareness (in the plain model). Informally, given a homomorphism-extractable encryption scheme for any class containing the constant functions $f \equiv m$, and where the evaluation algorithm is also function-hiding, it is possible to construct a plaintext aware encryption scheme as follows. The public-key of the new encryption scheme will be an encryption c_0 of 0, and an encryption of a message m is the homomorphic evaluation of the function $f \equiv m$ on c_0, the decryption algorithm is not modified. The homomorphism-extractability definition in this paper, would suffice for a relatively weak notion of plaintext awareness with a single decryption query, corresponding to one-time (or non-interactive) extraction, see [BP04].

4 Open Problems

Our work leaves open several questions that we believe are interesting to investigate.

Complexity-preserving PCPs. Theorem 1 gives a complexity-preserving MIP. Can one obtain (or rule out) complexity-preserving PCPs? Answering this question in the negative may run into difficult time-space tradeoff questions; still, the specificity of the setting might mean that such tradeoffs are easier to show.

Complexity preservation and public coins. Unlike Kilian's protocol, the interactive protocol we obtain in Theorem 3 is *not* of the public-coin type.[10] Is there an efficiency-preserving transformation from MIPs that yields a public-coin succinct argument? (For example, one way to do this would be to construct a public-coin succinct multi-function commitment.) More generally, are there public-coin complexity-preserving succinct arguments whose security can be based on standard cryptographic assumptions? (And can these be proved to in fact be universal arguments? Cf. Remark 3.5.)

Analogously, Theorem 4 only gives complexity-preserving SNARKs for designated verifiers. Is there a "direct" construction of a complexity-preserving publicly-verifiable SNARK? (I.e., without going through the generic transformation of [BCCT12b].)

Complexity preservation and random oracles. Can one obtain complexity-preserving (interactive or non-interactive) succinct arguments in the random-oracle model (with unconditional security)? We do not know of any candidate approaches for this question other than using complexity-preserving PCPs in a Kilian-type protocol; however, as already mentioned, we do not know of complexity-preserving PCPs. A related question is whether there are any succinct function commitments in the random-oracle model.

Power of succinct function commitments. We believe that succinct function commitments are a cryptographic primitive of independent interest. What other applications do they have? Also, our construction based on FHE from Theorem 2 supports *all* polynomial-time functions, while for our application we only need to be able to support multivariate polynomials of degree $\mathrm{polylog}(k)$, which may admit simpler (or

[10] An inconvenient consequence is the following: while the verifier's first message in our protocol is independent of the statement to be proven (as in Kilian's protocol), this message is not reusable across many statements.

more efficient) constructions. We note that existing works on polynomial commitment schemes or polynomial delegation schemes [KZG10, PST11, BGV11, PRV12, FG12] are not applicable here: the polynomial commitment schemes require a long preprocessing phase from the verifier, and the polynomial delegation schemes consider a different model where the verifier knows the polynomial beforehand and delegates it (in contrast, in our setting, the polynomial is some encoding of a witness that is only known to the prover). Another difference is that, in terms of security, we only require that the prover is committed to *some* function (not necessarily a low-degree polynomial).

5 Definitions

In this section we give basic definitions for the cryptographic primitives that we use (along with any non-standard properties that we may need). Throughout, $\mathrm{negl}(k)$ is any negligible function in k.

5.1 Homomorphic Encryption

An *$\mathcal{F}$-homomorphic encryption* scheme E is a tuple of algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$. The first three algorithms $\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}$ are the standard generation, encryption and decryption algorithms of a (semantically-secure) public-key encryption scheme. The additional algorithm Eval is a deterministic polynomial-time algorithm that takes as input a public key pk, a ciphertext $\mathsf{c} = \mathsf{Enc}_{\mathsf{pk}}(m)$, and (the representation of) a function $f \in \mathcal{F}$ and outputs a new ciphertext $\hat{\mathsf{c}} := \mathsf{Eval}_{\mathsf{pk}}(f, \mathsf{c})$ such that $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{c}) = f(m)$, where sk is the secret key corresponding to the public key pk. If $\mathcal{F}$ is the set of all polynomial-time functions, then we say that E is a *fully-homomorphic encryption* (FHE) scheme.

For homomorphic evaluation to be meaningful, Eval has to satisfy a certain *succinctness* requirement. Specifically, for any function $f \in \mathcal{F}$, with running time $t(n)$ and output size $\ell(n)$ on inputs of size n, and and any ciphertext c encrypting a plaintext of size n, (i) computing $\mathsf{Eval}(f, \mathsf{c})$ requires $\mathrm{poly}(k + t(n))$ time (cf. Remark 5.1 below) and (ii) its result $\hat{\mathsf{c}}$ has size $\mathrm{poly}(k + \ell(n))$ (which is, in particular, independent of the running time $t(n)$).

Remark 5.1 (complexity of homomorphic evaluation)**.** It is not known how to homomorphically evaluate a t-time random-access machine computation in less than time $t^2 \cdot \mathrm{poly}(k)$. Indeed, the homomorphic evaluation algorithm Eval usually consists of two "local algorithms", homomorphic addition $\mathsf{Eval}_{\mathsf{pk}}(+, \cdot)$ and homomorphic multiplication $\mathsf{Eval}_{\mathsf{pk}}(\times, \cdot)$ over $\mathbb{F}_2$, that are iteratively applied to the gates of the circuit to be evaluated. Thus, when we wish to homomorphically evaluate a random-access machine computation, we must first obtain a circuit representing such a computation; the problem is that it is not known how to reduce a t-time random-access machine computation to the evaluation of a circuit of size $o(t^2)$. (When the setting allows for the use of non-determinism, however, random-access machines do have small circuits; see, e.g., [BSCGT13].)

On the other hand, if we know "more" about f, then Eval might be able to run faster.

For example, if we know that f can be evaluated as a circuit in time t and space s (by some "gate-by-gate" evaluator algorithm), then we can bound the time and space complexity of $\mathsf{Eval}(f, \cdot)$ by $t \cdot \mathrm{poly}(k)$ and $s \cdot \mathrm{poly}(k)$ respectively. Whenever $s \ll t$ (for example, when the circuit to be evaluated corresponds to an algorithm using little memory and running for a long time), this better space bound is very attractive.

Finally, we note that what we mentioned in the previous paragraph also holds for *double* homomorphic evaluation: each of $\mathsf{Eval}_{\mathsf{pk}}(+, \cdot)$ and $\mathsf{Eval}_{\mathsf{pk}}(\times, \cdot)$ can be generically reduced (with a quadratic blow up) to a corresponding circuit, and it is not a problem to do so because the running time that we are squaring is only $\mathrm{poly}(k)$ (and is thus not dependent on t).

In light of Remark 5.1, we say that an FHE scheme is *local* if it has homomorphic addition $\mathsf{Eval}_{\mathsf{pk}}(+,\cdot)$ and homomorphic multiplication $\mathsf{Eval}_{\mathsf{pk}}(\times,\cdot)$ algorithms. Throughout this paper, we shall only consider local FHE schemes.[11]

Schemes satisfying the properties we need (including the local property) were recently constructed by, e.g., Gentry [Gen09], van Dijk et al. [vDGHV10], and Brakerski and Vaikuntanathan [BV11].

For simplicity, in our discussions we use *public-key* FHE schemes, but in our application we could also use secret-key ones. Up to some loss in efficiency, this is without loss of generality as the former can be constructed from the latter [Rot11].

5.2 Universal Relation and NP Relations

The *universal relation* [BG08] is defined to be the set $\mathcal{R}_{\mathcal{U}}$ of instance-witness pairs (y,w), where $y = (M,x,t)$, $|w| \leq t$, and M is an abstract machine, such that M accepts (x,w) after at most t steps. While the witness w for each instance $y = (M,x,t)$ is of size at most t, there is *no a-priori* polynomial bounding t in terms of $|x|$. We denote by $\mathcal{L}_{\mathcal{U}}$ the *universal language* corresponding to $\mathcal{R}_{\mathcal{U}}$. At high level, we can say that deciding membership in the universal relation of a given instance $y = (M,x,t)$ is a (non-deterministic) *bounded-halting problem* on the machine M.

For any $c \in \mathbb{N}$, we denote by $\mathcal{R}_c$ the subset of $\mathcal{R}_{\mathcal{U}}$ consisting of those pairs $(y,w) = \big((M,x,t),w\big)$ for which $t \leq |x|^c$; in other words, $\mathcal{R}_c$ is a "generalized" NP relation, where we do not insist on the same machine accepting different instances, but only insist on the same fixed polynomial bounding the running time in terms of the input size $|x|$. We denote by $\mathcal{L}_c$ the language corresponding to $\mathcal{R}_c$.

In this paper, we choose *random-access machines* [CR72, AV77] to be the abstract machine used in the universal relation $\mathcal{R}_{\mathcal{U}}$. This concrete choice is crucial in order to discuss the notion of complexity preservation for multi-prover interactive proofs (cf. Definition 5.3) and succinct arguments (cf. Definition 5.7 and Definition 5.12). For simplicity, in order avoid technicalities associated with how large is the transition function of a given random-access machine M when viewed as a circuit, we fix any "natural" universal architecture of the machine so that the only variation across different random-access machines is the sequence of basic instructions making up the program of the machine. Also, because we shall rely on a result of Ben-Sasson et al. [BSCGT13], for consistency with their result, we do not employ the logarithmic-cost criterion but instead count the number of state transitions of the machine and, in terms of space complexity, we measure the maximum number of memory cells used throughout the computation.

5.3 Succinct Multi-Prover Interactive Proofs Of Knowledge

In an *interactive proof* [GMR89, Bab85] a probabilistic polynomial-time verifier may interactively ask questions to a prover with unbounded computational power in order to decide the truth of a statement. These are a generalization of an "NP proof system", where the verifier does not ask any questions or toss coins, and only gets a single non-interactive proof from an unbounded prover. It is a celebrated result that IP, the class of all languages having an interactive proof, equals PSPACE [Sha92].

Ben-Or et al. [BOGKW88] generalized the setting of interactive proofs to one where instead of a single prover there are many. In a *multi-prover* interactive proof (MIP), a verifier V conducts a simultaneous interactive protocol with each of several provers, $P_1,\ldots,P_\ell$, that are assumed to not be able to communicate during the protocol. (But, of course, $P_1,\ldots,P_\ell$ may share correlated randomness.) It is known that MIP, the class of all languages having a multi-prover interactive proof, equals NEXP [BFL90].

[11] More generally, we require FHE schemes that are able to homomorphically evaluate a given circuit C in time $t \cdot \mathrm{poly}(k)$ and space $s \cdot \mathrm{poly}(k)$ whenever C can be evaluated gate-by-gate in time t and space s.

Vector notation. We use vector notation to denote multiple circuits each getting as input a different input as well as a shared input. For example, when we write "a circuit $\vec{A}$" we mean that $\vec{A}$ is a vector of (an unspecified number of) circuits $(A_i)_i$ (or $(A|_i)_i$ if $\vec{A}$ already has a subscript) and when we write "$\vec{z} = \vec{A}(\vec{x}, y)$" we mean that $z_i = A_i(x_i, y)$ for each i.[12] (We stress that A_i does not get to see x_j for $j \neq i$.)

Succinctness. While MIP protocols have found applications to program testing [BK89, BFL90] and approximation [FGL+91, AS98, ALM+98], they too, just like PCPs [BFLS91], can be used to enable a weak verifier to check long computations. Specifically, in a *succinct* MIP protocol, the verifier V runs in time $p(|y|)$ to check membership of an instance $y = (M, x, t)$ in the universal language $\mathcal{L}_{\mathcal{U}}$, when given the help of $\ell(|y|)$ provers each running in time $p(t)$, where p is a fixed universal polynomial (and ℓ is some "prover number" function).

Number of rounds. A *one-round* MIP is one where a verifier sends a single message q_i to each prover P_i, then each prover P_i replies with a message π_i, and finally the verifier accepts or rejects. In fact, we wish to notationally separate the process that generates messages from the process that verifies these and the provers' replies; we thus denote by G the (probabilistic) generator that produces the vector of messages $\vec{q}$ and denote by V the corresponding (deterministic) verification test. There are two technical details that we should address here:

- *What randomness does V see?* The probabilistic generator G tosses coins to obtain a random string r in order to generate the vector of messages $\vec{q}$; so we have the choice of giving r to V, or only $\vec{q}$. It will actually suffice to only give $\vec{q}$ to V, so that, overall, his inputs are $(\vec{q}, y, \vec{\pi})$.
- *What inputs does G get?* In principle, G may need $y = (M, x, t)$ as input in order to sample the vector of messages $\vec{q}$. It will actually suffice to only give t as input to G. We call such an MIP an *adaptive* MIP.

Knowledge and amplification. An MIP *of knowledge* is one where we are guaranteed that, when the verifier is convinced on input y, with probability above a certain *knowledge threshold* ε, by a vector of provers $\vec{P^*}$, an extractor E is able, by interacting with the provers $\vec{P^*}$, to extract a valid witness w for y. When constructing MIPs, we shall restrict our attention to achieving some constant *knowledge threshold*, e.g., $\varepsilon = 1/2$. The knowledge threshold can always be reduced to ε^k, by independently repeating the basic ℓ-protocol k times: the new prover would consist of k sets, each consisting of ℓ-provers. In case we are interested in knowledge threshold ε^k, all MIP algorithms would also get 1^k as input.

We now give the formal definition of (one-round) succinct MIPs of knowledge that we are interested in. As mentioned above, in the basic definition we will restrict attention to constant knowledge threshold $\varepsilon = 1/2$.

Definition 5.2. *A quadruple of algorithms (G, P, V, E) is a (one-round)* **succinct MIP of knowledge** *if the following conditions are satisfied:*

1. *Completeness*

   For every $(y, w) \in \mathcal{R}_{\mathcal{U}}$,

   $$\Pr_{\vec{q} \leftarrow G(t)} \left[V(\vec{q}, y, \vec{P}(\vec{q}, y, w)) = 1 \right] = 1 \ ,$$

   where each $P_i(\cdot)$ is defined as the algorithm $P(i, \cdot)$.

[12] Of course, a formal treatment here should be in terms of *families of vectors of circuits*: namely, $\vec{A}$ in fact denotes $\{\vec{A}_k\}_{k \in \mathbb{N}} = \{(A_{1,k}, \ldots, A_{\ell(k),k})\}_{k \in \mathbb{N}}$ for some "vector length function" $\ell \colon \mathbb{N} \to \mathbb{N}$ and $\vec{A}(\vec{x}, y)$ denotes the vector $(A_{1,k}(x_1, y), \ldots, A_{\ell(k),k}(x_{\ell(k)}, y))$ for the appropriate "input size" k. Furthermore, when discussing the efficiency of $\vec{A}$, one should also specify how uniform the family $\vec{A}$ is.

2. *Proof of knowledge*

   For every prover $\vec{P}^*$ and instance $y = (M, x, t)$,

   $$\Pr_{\vec{q} \leftarrow G(t)} \left[V(\vec{q}, y, \vec{P}^*(\vec{q}, y)) = 1 \right] > 1/2 \;\; \rightarrow \;\; \left(y, E^{\vec{P}^*}(y, 1^t) \right) \in \mathcal{R}_{\mathcal{U}} \;\; .$$

   Furthermore, E runs in polynomial time.

3. *Succinctness*

   There exists a fixed polynomial p such that for every instance $y = (M, x, t)$,

   - *the generator G runs in time $p(\log t)$;*
   - *each prover P_i runs in time $p(t)$;*
   - *the verifier V runs in time $p(|y|)$;*
   - *the communication transcript has size $p(\log t)$.*

A *complexity-preserving* succinct MIP of knowledge is a succinct MIP of knowledge where the prover time and space complexity, when proving the correctness of a nondeterministic random-access machine computation, is essentially the same as in the original computation:

Definition 5.3. *A quadruple of algorithms (G, P, V, E) is a* **complexity-preserving (one-round) succinct MIP of knowledge** *if it is a (one-round) succinct MIP of knowledge where the succinctness requirement is replaced by the following stronger requirement:*

Complexity-preserving succinctness

There exists a fixed polynomial p such that for every instance $y = (M, x, t)$ where M runs in space s,

- *the generator G runs in time $p(\log t)$;*
- *each prover P_i runs in time $(|M| + |x| + t) \cdot p(\log t)$;*
- *each prover P_i runs in space $(|M| + |x| + s) \cdot p(\log t)$;*
- *the verifier V runs in time $(|M| + |x| + \log t) \cdot p(\log t)$;*
- *the communication transcript has size $p(\log t)$.*

in the k-fold repetition of the protocol (meant to reduce the knowledge threshold), all the above measures may increase upto a factor k.

Remark 5.4. When $\mathcal{R} \in \mathsf{NP}$ there is no need to provide a time bound t to G: we will work with some arbitrary polynomial bound $t \leq k^{\log k}$. In such a case, we will simply write $G(1^k)$.

Remark 5.5 (local witness decoding)**.** In Definition 5.2, we can consider a stronger notion of proof of knowledge that requires "local extraction" [BG08, Definition 3.2]:

- There exists a local knowledge extractor $\hat{E}$ such that, for every prover $\vec{P}^*$ and instance y,

  $$\Pr_{\vec{q} \leftarrow G(t)} \left[V(\vec{q}, y, \vec{P}^*(\vec{q}, y)) = 1 \right] > 1/2 \;\; \rightarrow \;\; \exists\, w \text{ s.t. } \begin{array}{l} (y, w) \in \mathcal{R}_{\mathcal{U}} \text{ and} \\ \Pr \left[\hat{E}^{\vec{P}^*}(y, i) = w_i \right] > \frac{2}{3} \end{array} \;\; .$$

Ultimately, our constructions will satisfy these slightly stronger definitions; see Section 6 for details.

5.4 Succinct Arguments Of Knowledge

A succinct argument [BG08] is a pair of algorithms $(\mathcal{P}, \mathcal{V})$ that works as follows. Given an instance $y = (M, x, t)$, a prover $\mathcal{P}$ (who is also given as auxiliary input a witness w proving that $y \in \mathcal{L}_{\mathcal{U}}$) and a verifier $\mathcal{V}$ engage in interactive protocol, at the end of which $\mathcal{V}$ either accepts or rejects.

Definition 5.6. *A pair of algorithms $(\mathcal{P}, \mathcal{V})$ is a* **succinct argument** *for the relation $\mathcal{R} \subseteq \mathcal{R}_{\mathcal{U}}$ if the following conditions are satisfied:*

1. *Completeness*

   For every $(y, w) \in \mathcal{R}_{\mathcal{U}}$ and $k \in \mathbb{N}$,

   $$\Pr\left[\langle \mathcal{P}(w), \mathcal{V}\rangle(1^k, y) = 1\right] = 1 \ .$$

2. *Soundness*

   For every polynomial-size prover $\mathcal{P}^*$, instance $y \notin \mathcal{L}_{\mathcal{R}}$, and large enough security parameter $k \in \mathbb{N}$,

   $$\Pr\left[\langle \mathcal{P}^*, \mathcal{V}\rangle(1^k, y) = 1\right] \leq \mathrm{negl}(k) \ .$$

3. *Succinctness*

   There exists a universal polynomial p (i.e., independent of $\mathcal{R}$) such that for every instance $y = (M, x, t)$ and large enough security parameter $k \in \mathbb{N}$,

   - *the prover $\mathcal{P}$ runs in time $p(k + t)$;*
   - *the verifier $\mathcal{V}$ runs in time $p(k + |y|)$;*
   - *the communication transcript has size $p(k + \log t)$.*

A *complexity-preserving* succinct argument is a succinct argument where the prover and verifier complexities are essentially optimal when proving and verifying the correctness of nondeterministic random-access machine computations:

Definition 5.7. *A pair of algorithms $(\mathcal{P}, \mathcal{V})$ is a* **complexity-preserving succinct argument** *for the relation $\mathcal{R} \subseteq \mathcal{R}_{\mathcal{U}}$ if it is a succinct argument for $\mathcal{R}$ where succinctness is replaced by the following stronger requirement:*

Complexity-preserving succinctness

There exists a universal polynomial p (i.e., independent of $\mathcal{R}$) such that for every instance $y = (M, x, t)$ where M runs in space s and large enough security parameter $k \in \mathbb{N}$,

- *the prover $\mathcal{P}$ runs in time $(|M| + |x| + t) \cdot p(k + \log t)$;*
- *the prover $\mathcal{P}$ runs in space $(|M| + |x| + s) \cdot p(k + \log t)$;*
- *the verifier $\mathcal{V}$ runs in time $(|M| + |x| + \log t) \cdot p(k + \log t)$;*
- *the communication transcript has size $p(k + \log t)$.*

A *succinct argument of knowledge* is a succinct argument where soundness is strengthened as follows:

Definition 5.8. *A pair of algorithms $(\mathcal{P}, \mathcal{V})$ is a* **succinct argument of knowledge** *for the relation $\mathcal{R} \subseteq \mathcal{R}_{\mathcal{U}}$ if it is a succinct argument for $\mathcal{R}$ where soundness is replaced by the following stronger requirement:*

Proof of knowledge

For every two positive polynomials s and p there exist a positive polynomial q and a probabilistic polynomial-time knowledge extractor $\mathcal{E}$ such that, for every family of s-size prover circuits $\mathcal{P}^*$, every instance $y = (M, x, t)$, and every sufficiently large $k \in \mathbb{N}$, the following holds:

Suppose that the prover circuit $\mathcal{P}^*_k$ convinces $\mathcal{V}$ to accept y with probability greater than $p(k)^{-1}$ (taken over a random choice of internal randomness for $\mathcal{V}$).

Then, with probability greater than $q(k)^{-1}$ taken over a random choice of internal randomness r for $\mathcal{E}$, the knowledge extractor $\mathcal{E}$, with oracle access to the code of $\mathcal{P}^*_k$ and on input $(y, 1^t)$, outputs a valid witness w for y,

In symbols:

$$\forall\, s\ \forall\, p\ \exists\, q\ \exists\, \mathcal{E}\ \forall\, s\text{-size } \mathcal{P}^*\ \exists\, K\ \forall\, k > K\ \forall\, y = (M, x, t)$$

$$\Pr\Big[\langle \mathcal{P}^*_k, \mathcal{V}\rangle(1^k, y) = 1\Big] > \frac{1}{p(k)} \quad \longrightarrow \quad \Pr_r\Big[\, w \in \mathcal{R}_{\mathcal{U}}(y) \,\Big|\, w \leftarrow \mathcal{E}^{\mathcal{P}^*_k}(1^k, y, 1^t\,; r)\,\Big] > \frac{1}{q(k)}\ . \quad (1)$$

Remark 5.9 (local witness decoding)**.** Barak and Goldreich [BG08] define *universal arguments* to be succinct arguments of knowledge for $\mathcal{R} = \mathcal{R}_{\mathcal{U}}$ as in Definition 5.8, with a proof of knowledge property where the extractor is an *implicit* representation of the witness. The only difference is that Equation (1) is replaced by the following one:

$$\Pr\Big[\langle \mathcal{P}^*_k, \mathcal{V}\rangle(1^k, y) = 1\Big] > \frac{1}{p(k)}$$
$$\longrightarrow\ \Pr_r\Big[\, \exists w = w_1 \cdots w_t \in \mathcal{R}_{\mathcal{U}}(y) \text{ s.t. } \forall\, i \in [t]\,,\, \mathcal{E}^{\mathcal{P}^*_k}(1^k, y, i\,; r) = w_i\,\Big] > \frac{1}{q(k)}\ .$$

Our constructions will achieve this stronger property.

5.5 SNARKs

A *succinct non-interactive argument* (SNARG) is a triple of algorithms $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ that works as follows. The (probabilistic) generator $\mathcal{G}$, on input the security parameter k, outputs a *reference string* σ and a corresponding *verification state* τ. The honest prover $\mathcal{P}(y, w, \sigma)$ produces a proof π for the statement $y = (M, x, t)$ given a witness w; then $\mathcal{V}(\tau, y, \pi)$ verifies the validity of π. The SNARG is *adaptive* if the prover may choose the statement *after* seeing the σ, otherwise, it is *non-adaptive*.

Definition 5.10. *A triple of algorithms $(\mathcal{P}, \mathcal{G}, \mathcal{V})$ is a SNARG for the relation $\mathcal{R} \subseteq \mathcal{R}_{\mathcal{U}}$ if the following conditions are satisfied:*

1. *Completeness*

   For every large enough security parameter $k \in \mathbb{N}$, every time bound $T \in \mathbb{N}$, and every $(y, w) = \big((M, x, t), w\big) \in \mathcal{R}$ with $t \leq T$,

$$\Pr_{(\sigma,\tau)\leftarrow\mathcal{G}(1^k,T)}\Big[\mathcal{V}(\tau, y, \pi) = 1 \,\Big|\, \pi \leftarrow \mathcal{P}(\sigma, y, w)\Big] = 1\ .$$

2. *Soundness (depending on which notion is considered)*

   - ***Non-adaptive soundness.*** *For every polynomial-size prover $\mathcal{P}^*$, every large enough security parameter $k \in \mathbb{N}$, every time bound $T \in \mathbb{N}$, and every instance $y = (M, x, t) \notin \mathcal{L}_{\mathcal{U}}$ with $t \leq T$,*

$$\Pr_{(\sigma,\tau)\leftarrow\mathcal{G}(1^k,T)}\left[\mathcal{V}(\tau, y, \pi) = 1 \;\middle|\; \pi \leftarrow \mathcal{P}^*(\sigma, y)\right] \leq \mathrm{negl}(k) \ .$$

   - ***Adaptive soundness.*** *For every polynomial-size prover $\mathcal{P}^*$, every large enough security parameter $k \in \mathbb{N}$, and every time bound $T \in \mathbb{N}$,*

$$\Pr_{(\sigma,\tau)\leftarrow\mathcal{G}(1^k,T)}\left[\begin{array}{c} t \leq T \\ \mathcal{V}(\tau, y, \pi) = 1 \\ y \notin \mathcal{L}_{\mathcal{U}} \end{array} \;\middle|\; (y, \pi) \leftarrow \mathcal{P}^*(\sigma)\right] \leq \mathrm{negl}(k) \ .$$

3. *Succinctness*

   There exists a universal polynomial p (i.e., independent of $\mathcal{R}$) such that for every instance $y = (M, x, t)$ and large enough security parameter $k \in \mathbb{N}$,

   - *the generator $\mathcal{G}(1^k, t)$ runs in time $p(k + \log t)$;*
   - *the prover $\mathcal{P}$ runs in time $p(k + t)$;*
   - *the verifier $\mathcal{V}$ runs in time $p(k + |y|)$;*
   - *an honestly generated proof has size $p(k + \log t)$.*

Remark 5.11. When $\mathcal{R} \in \mathsf{NP}$ there is no need to provide a time bound T to $\mathcal{G}$: we can use $\mathcal{G}(1^k, k^{\log k})$. In such a case, we simply write $\mathcal{G}(1^k)$ and $\mathcal{G}$ runs in a fixed polynomial in k.

A *complexity-preserving* SNARG is a SNARG where the prover and verifier complexities are essentially optimal when proving and verifying the correctness of nondeterministic random-access machine computations:

Definition 5.12. *A triple of algorithms $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is a* **complexity-preserving SNARG** *for the relation $\mathcal{R} \subseteq \mathcal{R}_{\mathcal{U}}$ if it is a SNARG for $\mathcal{R}$ where succinctness is replaced by the following stronger requirement:*

Complexity-preserving succinctness

There exists a universal polynomial p (i.e., independent of $\mathcal{R}$) such that for every instance $y = (M, x, t)$ where M runs in space s and large enough security parameter $k \in \mathbb{N}$,

- *the generator $\mathcal{G}(1^k, t)$ runs in time $p(k + \log t)$;*
- *the prover $\mathcal{P}$ runs in time $(|M| + |x| + t) \cdot p(k + \log t)$;*
- *the prover $\mathcal{P}$ runs in space $(|M| + |x| + s) \cdot p(k + \log t)$;*
- *the verifier $\mathcal{V}$ runs in time $(|M| + |x| + \log t) \cdot p(k + \log t)$;*
- *an honestly generated proof has size $p(k + \log t)$.*

A *SNARG of knowledge*, or SNARK for short, is a SNARG where soundness is strengthened as follows:

Definition 5.13. *A triple of algorithms $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is a SNARK for $\mathcal{R} \subseteq \mathcal{R}_{\mathcal{U}}$ if it is a SNARG for $\mathcal{R}$ where adaptive soundness is replaced by the following stronger requirement:*

- *Adaptive proof of knowledge*[13]

  For every polynomial-size prover $\mathcal{P}^*$ there exists a polynomial-size extractor $\mathcal{E}_{\mathcal{P}^*}$ such that for every large enough security parameter $k \in \mathbb{N}$, every auxiliary input $z \in \{0,1\}^{\mathrm{poly}(k)}$, and time bound $T \in \mathbb{N}$,

$$\Pr_{(\sigma,\tau)\leftarrow\mathcal{G}(1^k,T)}\left[\begin{array}{c|c} t \le T & (y,\pi)\leftarrow\mathcal{P}^*(z,\sigma) \\ \mathcal{V}(\tau,y,\pi)=1 & w\leftarrow\mathcal{E}_{\mathcal{P}^*}(z,\sigma) \\ w\notin\mathcal{R}(y) & \end{array}\right] \le \mathrm{negl}(k) \ .$$

Remark 5.14 (public verification vs. designated verifiers)**.** One may want to distinguish between the case where the verifier state τ is allowed to be public or needs to remain private. Specifically, a *publicly-verifiable SNARK* (pvSNARK) is one where security holds even if τ is public; a *designated-verifier SNARK* (dvSNARK) is one where τ needs to remain secret. In this paper, we will only be concerned with dvSNARKs.

We note that a very desirable property is the ability to generate σ once and for all and then reuse it in polynomially-many proofs (potentially by different provers). For pvSNARKs, this *multi-theorem soundness* is automatically guaranteed; for dvSNARKs, however, multi-theorem soundness needs to be required explicitly as an additional property. Usually, this is achieved by ensuring that the verifier's response "leaks" only a negligible amount of information about τ.[14] In this paper we will construct dvSNARKs that do *not* satisfy multi-theorem soundness.

Remark 5.15 (assumptions on the reference string)**.** Depending on the model and required properties, there may be different trust assumptions about who runs $\mathcal{G}(1^k)$ to obtain (σ,τ), publish σ, and make sure the verifier has access to τ.

For example, in a dvSNARK, the verifier itself may run $\mathcal{G}$ and then publish σ (or send it to the appropriate prover when needed) and keep τ secret for later; in such a case, we may think of σ as a *verifier-generated* reference string. But if we further require the property of *zero-knowledge*, then we must assume that σ is a *common* reference string, and thus that a trusted party ran $\mathcal{G}$.

As another example, in a pvSNARK, the verifier may once again run $\mathcal{G}$ and then publish σ, though other verifiers if they do not trust him may have to run on their own $\mathcal{G}$ to obtain outputs that they trust; alternatively, we could assume that σ is a *global reference string* that everyone trusts. Once again, if we further require the property of zero-knowledge, then we must assume that σ is a *common* reference string.

Remark 5.16 (universal arguments vs. weaker notions)**.** A SNARK for the relation $\mathcal{R} = \mathcal{R}_{\mathcal{U}}$ is called a *universal argument*.[15] A weaker notion that we will also consider is a SNARK for the relation $\mathcal{R} = \mathcal{R}_c$ for a constant $c \in \mathbb{N}$. In this case, soundness is only required to hold with respect to $\mathcal{R}_c$; in particular, the verifier algorithm may depend on c . Nevertheless, we require (and achieve) *universal succinctness*, where a universal polynomial p, independent of c, upper bounds the length of every proof and the verification time.

Remark 5.17 (on the SNARK extractor)**.** In Definition 5.13 we require that any polynomial-size family of circuits $\mathcal{P}^*$ has a specific polynomial-size family of extractors $\mathcal{E}_{\mathcal{P}^*}$; in particular, we allow the extractor to be of arbitrary polynomial-size and to be "more non-uniform" than $\mathcal{P}^*$. In addition, we require that for

[13] One can also formulate weaker proof of knowledge notions; in this work we focus on the above strong notion.

[14] Note that $O(\log k)$-theorem soundness always holds; the "non-trivial" case is whenever $\omega(\log k)$. Weaker solutions to support more theorems include simply assuming that the verifier's responses remain secret, or re-generating σ every logarithmically-many rejections, e.g., as in [KR06, Mie08, GKR08, KR09, GGP10, CKV10].

[15] Barak and Goldreich [BG08] define universal arguments for $\mathcal{R}$ with a black-box "weak proof-of-knowledge" property (cf. Remark 5.9); in contrast, our proof of knowledge property is not restricted to black-box extractors, and does not require the extractor to be an implicit representation of a witness.

any prover auxiliary input $z \in \{0,1\}^{\mathrm{poly}(k)}$, the polynomial-size extractor manages to perform its witness-extraction task given the same auxiliary input z. The definition can be naturally relaxed to consider only specific distributions of auxiliary inputs according to the required application. (In our setting, the restrictions on the auxiliary input handled by the knowledge extractor will be related to the auxiliary input that the underlying HEE extractor can handle. See further discussion in Section 9.1.)

One could consider stronger notions in which the extractor is a uniform machine that gets $\mathcal{P}^*$ as input, or even only gets black-box access to $\mathcal{P}^*$. (For the case of adaptive SNARKs, this notion cannot be achieved based on black-box reductions to falsifiable assumptions [GW11].) In common security reductions, however, where the primitives (to be broken) are secure against arbitrary polynomial-size non-uniform adversaries, the non-uniform notion seems to suffice. In our case, going from a knowledge assumption to SNARKs, the notion of extraction will be preserved: if you start with uniform extraction you will get SNARK with uniform extraction.

Remark 5.18 (extraction for multiple theorems)**.** We will often be interested in provers that may produce a vector of theorems $\vec{y}$ together with a vector of corresponding proofs $\vec{\pi}$. In such cases, it will be more convenient to apply corresponding extractors that can extract a vector of witnesses $\vec{w}$ for each of the proofs that is accepted by the SNARK verifier. This notion of extraction can be shown to follow from the single-witness extraction as defined above [BCCT12a].

Lemma 5.19 (adaptive proof of knowledge for multiple inputs [BCCT12a])**.** *Let $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ be a SNARK (as in Definition 5.13), then for any polynomial-size multi-theorem prover $\mathcal{P}^*$ there exists a polynomial-size extractor $\mathcal{E}_{\mathcal{P}^*}$ such that for every large enough security parameter $k \in \mathbb{N}$, every auxiliary input $z \in \{0,1\}^{\mathrm{poly}(k)}$, and every time bound $T \in \mathbb{N}$,*

$$\Pr_{(\sigma,\tau)\leftarrow\mathcal{G}(1^k,T)}\left[\begin{array}{c|c} \exists\, i \ \text{ s.t. } \mathcal{V}(\tau, y_i, \pi_i) = 1 \\ \ \ w_i \notin \mathcal{R}(y_i) \end{array} \begin{array}{c} t_i \le T \\ (\vec{y},\vec{\pi}) \leftarrow \mathcal{P}^*(z,\sigma) \\ (\vec{y},\vec{w}) \leftarrow \mathcal{E}_{\mathcal{P}^*}(z,\sigma) \end{array}\right] \le \mathrm{negl}(k) \ .$$

6 A Complexity-Preserving 1-Round Succinct MIP Of Knowledge

We prove Theorem 1, discussed in Section 3.1. A two-step construction outline was given in Section 3.1. Sections 6.1, 6.2, 6.3 achieve the first step of that construction outline; while Sections 6.4 and 6.5 achieve the second step. Throughout, it will be important to keep the definitions from Section 5.3 in mind.

Basic notations for this section. Given a finite alphabet Σ and $N \in \mathbb{N}$, for any Σ-strings a and b in Σ^N, the *fractional (Hamming) distance* over the alphabet Σ between a and b, denoted $\Delta_N(a,b)$, is equal to the number of positions where a and b differ divided by N.

Let $\mathbb{F}$ be a field, S a subset of $\mathbb{F}$, $d \in \mathbb{N}$ a degree, and $m \in \mathbb{N}$ a dimension. We denote by $\mathrm{RM}(\mathbb{F}, S, d, m)$ the set of functions $p\colon S^m \to \mathbb{F}$ that are evaluations of m-variate polynomials of at most (total) degree d. (Of course, RM stands for Reed–Muller.) For a subset H of $\mathbb{F}$, we denote by $\mathrm{VRM}(\mathbb{F}, S, H, d, m)$ the subcode of $\mathrm{RM}(\mathbb{F}, S, d, m)$ consisting of all evaluations of polynomials that *vanish* on H^m.

Recall the standard notion of low-degree extension:

Definition 6.1. *Let $\mathbb{F}$ be a field, H a subset of $\mathbb{F}$, and $m \in \mathbb{N}$ a dimension. The* **low-degree extension** *of a function $f\colon H^m \to \mathbb{F}$ is the polynomial over $\mathbb{F}$ of degree less than $|H|$ in each variable given by the following expression:*

$$\hat{f}(z_1,\ldots,z_m) := \sum_{\vec{\alpha}\in H^m} f(\vec{\alpha}) \cdot L_{\mathbb{F},H,m,\vec{\alpha}}(\vec{z}) \ ,$$

where $L_{\mathbb{F},H,m,\vec{\alpha}}(\vec{z}) := \prod_{i=1}^{m} \prod_{\beta \in (H - \{\alpha_i\})} \frac{z_i - \beta}{\alpha_i - \beta}$.

6.1 An MIP For Testing Proximity To RM

We begin by considering the problem of testing multivariate polynomials for low degree, that is, the problem of testing proximity to RM.

Moshkovitz and Raz [MR08] gave a detailed and more general analysis of the Subspace vs. Point Tester of Raz and Safra [RS97]. We show how to convert this tester to obtain an MIP protocol with 2 provers for testing proximity to RM.

Let us briefly recall the Subspace vs Point Tester. The tester gets oracle access to two functions:

- a function $f \colon \mathbb{F}^m \to \mathbb{F}$, which is the one whose proximity to RM we wish to test, and
- a proximity proof $\Pi \colon \mathbb{F}^m \times \mathbb{H}^m \times \mathbb{H}^m \to \mathbb{F}^{d^3}$, which, for every three points $\vec{\alpha}, \vec{\beta}_1, \vec{\beta}_2$ with $\vec{\alpha} \in \mathbb{F}^m$ and $\vec{\beta}_1, \vec{\beta}_2 \in \mathbb{H}^m$, gives a polynomial of degree at most d corresponding to the (typically three-dimensional) subspace in $\mathbb{F}^m$ spanned by $\vec{\alpha}, \vec{\beta}_1, \vec{\beta}_2$.

The tester works as follows:

$\textsc{SpacePoint}^{f,\Pi}_{\mathbb{F},\mathbb{H},m,d} \stackrel{\text{def}}{=}$

1. Draw uniformly and independently at random: $\vec{\alpha} \leftarrow \mathbb{F}^m$ and $\vec{\beta}_1, \vec{\beta}_2 \leftarrow \mathbb{H}^m$.
2. Query f at $\vec{\alpha}$ to obtain $\omega \in \mathbb{F}$.
3. Query Π at $(\vec{\alpha}, \vec{\beta}_1, \vec{\beta}_2)$ to obtain a polynomial Q' of degree at most d. (Corresponding to the subspace defined by the vectors $(\vec{\alpha}, \vec{\beta}_1, \vec{\beta}_2)$.)
4. Accept if and only if $\omega = Q'(\vec{\alpha})$ or $\vec{\alpha}, \vec{\beta}_1, \vec{\beta}_2$ are linearly dependent.

The testing guarantees are as follows:

Theorem 6.2 ([MR08])**.** *Let $\mathbb{F}$ be a field, $\mathbb{H}$ a subspace of $\mathbb{F}$, $m \in \mathbb{N}$ a dimension, and $d \in \mathbb{N}$ a degree. Define $\varepsilon \stackrel{\text{def}}{=} 2^7 m\left(\frac{1}{|\mathbb{H}|^8} + (\frac{md}{|\mathbb{F}|})^4\right)$. For every function $f \colon \mathbb{F}^m \to \mathbb{F}$ and proximity proof $\Pi \colon \mathbb{F}^m \times \mathbb{H}^m \times \mathbb{H}^m \to \mathbb{F}^{d^3}$, and every success probability*

$$\gamma \stackrel{\text{def}}{=} \Pr_{\substack{\vec{\alpha} \leftarrow \mathbb{F}^m \\ \vec{\beta}_1, \vec{\beta}_2 \leftarrow \mathbb{H}^m}} \left[\textsc{SpacePoint}^{f,\Pi}_{\mathbb{F},\mathbb{H},m,d}(\vec{\alpha}, \vec{\beta}_1, \vec{\beta}_1) = 1 \right] > 0 \ ,$$

there exists a polynomial $Q \colon \mathbb{F}^m \to \mathbb{F}$ with $\deg(Q) \leq d$ such that

$$\Pr_{\vec{z} \leftarrow \mathbb{F}^m} \left[Q(\vec{z}) = f(\vec{z}) \right] \geq \gamma - 3\varepsilon \ .$$

We can obtain an MIP verifier V_{RM} for testing proximity to RM by using two provers:

- the first prover P_1^* "represents" f, and
- the second prover P_2^* "represents" Π.

Namely, we define V_{RM} to run $\textsc{SpacePoint}^{f,\Pi}_{\mathbb{F},\mathbb{F},m,d}$ (that is, we choose $\mathbb{H} = \mathbb{F}$), by sending the query for f to P_1^* and the query for Π to P_2^*, and accept if and only if the tester does. The honest prover P_{RM} is then naturally defined and can convince V_{RM} to accept with probability 1 a polynomial $Q(z_1, \ldots, z_m)$ of degree at most d. More precisely, we define V_{RM} and P_{RM} as follows.

$V_{\mathrm{RM}}^{P_1^*,P_2^*}(\mathbb{F},m,d) \stackrel{\mathrm{def}}{=}$

1. Draw $\vec{\alpha},\vec{\beta}_1,\vec{\beta}_2 \leftarrow \mathbb{F}^m$.
2. Query P_1^* at $\vec{\alpha}$ to obtain $\omega \in \mathbb{F}$.
3. Query P_2^* at $(\vec{\alpha},\vec{\beta}_1,\vec{\beta}_2)$ to obtain a polynomial Q' of degree at most d on the subspace defined by the vectors $(\vec{\alpha},\vec{\beta}_1,\vec{\beta}_2)$.
4. Accept if and only if $\omega = Q'(\vec{\alpha})$ or $\vec{\alpha},\vec{\beta}_1,\vec{\beta}_2$ are linearly dependent.

$P_{\mathrm{RM}}(\mathbb{F},m,d,i,Q;q) \stackrel{\mathrm{def}}{=}$

1. If $i=1$:
   (a) parse q as a query $\vec{\alpha} \in \mathbb{F}^m$;
   (b) compute and output $Q(\vec{\alpha})$.
2. If $i=2$:
   (a) parse q as a query $(\vec{\alpha},\vec{\beta}_1,\vec{\beta}_2) \in \mathbb{F}^{3m}$;
   (b) compute and output the coefficients of the polynomial Q when restricted to the subspace spanned by $(\vec{\alpha},\vec{\beta}_1,\vec{\beta}_2)$.

By invoking Theorem 6.2, we can see that V_{RM} lets us test proximity to RM of the function induced by P_1^*:

Lemma 6.3. *Let $\mathbb{F}$ be a field, $m \in \mathbb{N}$ a dimension, and $d \in \mathbb{N}$ a degree. Define*

$$\varepsilon_{|\mathbb{F}|,m,d} \stackrel{\mathrm{def}}{=} \frac{2^7 m}{|\mathbb{F}|^8}\big(1+(m|\mathbb{F}|d)^4\big) \tag{2}$$

and

$$s^{\mathrm{RM}}_{|\mathbb{F}|,m,d}(\delta) := \delta - 3\cdot\varepsilon_{|\mathbb{F}|,m,d} \ . \tag{3}$$

For any two provers P_1^* and P_2^* the following holds:

$$\Pr\Big[V_{\mathrm{RM}}^{P_1^*,P_2^*}(\mathbb{F},m,d)=1\Big] \le 1 - s^{\mathrm{RM}}_{|\mathbb{F}|,m,d}\Big(\Delta\big(f_{P_1^*},\mathrm{RM}(\mathbb{F},\mathbb{F},m,d)\big)\Big) \ ,$$

where $f_{P_1^*}:\mathbb{F}^m \to \mathbb{F}$ is defined by $f_{P_1^*}(\vec{\alpha}) := P_1^*(\vec{\alpha})$ for every $\vec{\alpha} \in \mathbb{F}^m$.

Proof. Because the two provers cannot communicate, and because we only send a single query to P_1^* and a single query to P_2^*, each prover is actually equivalent to a corresponding string (over an appropriate alphabet), and we can thus directly invoke Theorem 6.2 to establish the lemma. □

Remark 6.4 (efficiency of V_{RM})**.** In terms of efficiency, note that V_{RM} only needs to sample three points, check for linear independence, and evaluate a trivariate polynomial of degree at most d; all of this can be done in $\mathrm{poly}(m+d)$ field operations.

Remark 6.5 (efficiency of P_{RM})**.** The honest prover $P_{\mathrm{RM}}|_{i=1}$ only needs to evaluate the given polynomial Q of degree d at a single point. The honest prover $P_{\mathrm{RM}}|_{i=2}$ only needs to find the coefficients of Q when restricted to a three-dimensional subspace; this can be done by evaluating Q on the entire subspace (i.e., on $|\mathbb{F}|^3$ points) and then interpolating. Overall, the prover's (time and space) complexity is dominated by having to evaluate Q at $\mathrm{poly}(|\mathbb{F}|)$ points.

Remark 6.6 (length of proximity proof)**.** Note that the proximity proof Π consists of $|\mathbb{F}^m| \cdot |\mathbb{H}^m|^2$ field elements. We can see here an example of a proof length versus soundness tradeoff typical of PCP constructions: making $\mathbb{H}$ a smaller subspace will decrease the proximity proof length but doing so will also worsen soundness. If we wanted the "best" soundness guaranteed by the test, we could set $\mathbb{H} = \mathbb{F}$, but then the proximity proof would have length that is *cubic* in $|\mathbb{F}^m|$ — extremely long!

And, indeed, in our MIP verifier V_{RM} we do set $\mathbb{H} = \mathbb{F}$, but we do not "pay" in proof length: the honest P_{RM} for $i=2$, which is "assigned" to behave according to the proximity proof, can compute on-the-fly the response to the verifier's specific query, without having to write down a priori all the possible answers!

6.2 An MIP For Testing Proximity To VRM

Next, we consider the problem of testing vanishing properties of low-degree multivariate polynomials, that is, the problem of testing proximity to VRM.

Ben-Sasson and Sudan [BSS08] showed how to generically transform a proximity tester for RM into a proximity tester for VRM [BSS08, Lemma 4.5]. We follow their construction to obtain an MIP protocol with $2(m+1)$ provers for testing proximity to VRM.

The main tool of the transformation is the following lemma:

Lemma 6.7 ([BSS08, Lemma 4.9])**.** *Let $\mathbb{F}$ be a field, H a subset of $\mathbb{F}$, $m \in \mathbb{N}$ a dimension, and $d \in \mathbb{N}$ a degree. A polynomial $Q\colon \mathbb{F}^m \to \mathbb{F}$ of degree d over $\mathbb{F}$ vanishes on H^m if and only if there exist polynomials $Q_1, \ldots, Q_m\colon \mathbb{F}^m \to \mathbb{F}$ of total degree at most d each such that*[16]

$$Q(z_1, \ldots, z_m) = \sum_{i=1}^{m} Z_H(z_i) Q_i(z_1, \ldots, z_m) \ , \tag{4}$$

where $Z_H(z) = \prod_{\gamma \in H}(z - \gamma)$.[17]

Given the MIP verifier V_{RM} for testing proximity to RM that we constructed in Section 6.1, we can now construct an MIP verifier V_{VRM} for testing proximity to VRM by using $2(m+1)$ provers:

- the first $(m+1)$ provers $P^*_1, \ldots, P^*_{m+1}$ respectively provide access to an evaluation $q\colon \mathbb{F}^m \to \mathbb{F}$ of (allegedly) the polynomial Q and evaluations $q_1, \ldots, q_m\colon \mathbb{F}^m \to \mathbb{F}$ of (allegedly) the m polynomials $Q_1, \ldots, Q_m$, and
- the second $(m+1)$ provers $P^*_{m+2}, \ldots, P^*_{2(m+1)}$ respectively provide access to corresponding $(m+1)$ proximity proofs to RM.

In this way, V_{VRM} can test proximity of q and $q_1, \ldots, q_m$ to RM and then perform a consistency check to verify Equation (4). While it is possible to construct V_{VRM} in a black-box way from V_{RM}, a simpler construction is to "open up" V_{RM}. The honest prover P_{VRM} is then naturally defined and can convince V_{VRM} to accept with probability 1 a polynomial $Q(z_1, \ldots, z_m)$ of degree at most d vanishing on H^m. More precisely:

$V_{\mathrm{VRM}}^{P^*_1, \ldots, P^*_{2(m+1)}}(\mathbb{F}, H, m, d) \stackrel{\mathrm{def}}{=}$

1. Draw $\vec{\alpha}, \vec{\beta}_1, \vec{\beta}_2 \leftarrow \mathbb{F}^m$.
2. For $i = 0, \ldots, m$:
   (a) query P^*_{i+1} at $\vec{\alpha}$ to obtain ω_i;
   (b) query $P^*_{(m+1)+i+1}$ at $(\vec{\alpha}, \vec{\beta}_1, \vec{\beta}_2)$ to obtain a trivariate polynomial Q_i of degree at most d.
3. Compute $Z_H(z) = \prod_{\gamma \in H}(z - \gamma)$.
4. Accept if and only if $\vec{\alpha}, \vec{\beta}_1, \vec{\beta}_2$ are linearly dependent, or $\omega_0 = Q_0(\vec{\alpha}), \ldots, \omega_m = Q_m(\vec{\alpha})$ and $\omega_0 = \sum_{i=1}^{m} Z_H(\alpha_i)\omega_i$.

$P_{\mathrm{VRM}}(\mathbb{F}, H, m, d, i, Q; q) \stackrel{\mathrm{def}}{=}$

1. If $i \in \{1, \ldots, m+1\}$:
   (a) parse q as a query $\vec{\alpha} \in \mathbb{F}^m$;
   (b) if $i = 1$, compute and output $Q(\vec{\alpha})$;
   (c) otherwise, compute and output $Q_{i-1}(\vec{\alpha})$.
2. If $i \in \{(m+1)+1, \ldots, 2(m+1)\}$:
   (a) parse q as a query $(\vec{\alpha}, \vec{\beta}_1, \vec{\beta}_2) \in \mathbb{F}^{3m}$;
   (b) if $i = 1$, compute and output the coefficients of the polynomial Q when restricted to the subspace spanned by $(\vec{\alpha}, \vec{\beta}_1, \vec{\beta}_2)$;
   (c) otherwise, compute and output the coefficients of the polynomial Q_{i-1} when restricted to the subspace spanned by $(\vec{\alpha}, \vec{\beta}_1, \vec{\beta}_2)$.

[16] In fact total degree $d - |H|$ would suffice.

[17] Actually, [BSS08, Lemma 4.9] is stated for the degree in each variable, but the same proof gives the same result for total degree, which is the one we are interested in.

The testing guarantees of V_{VRM} are given by the following lemma:

Lemma 6.8. *Let $\mathbb{F}$ be a field, H a subset of $\mathbb{F}$, $m \in \mathbb{N}$ be a dimension, and $d \in \mathbb{N}$ a degree. Define*

$$s^{\mathrm{VRM}}_{|\mathbb{F}|,|H|,m,d}(\delta) := \min\left\{ s^{\mathrm{RM}}_{|\mathbb{F}|,m,d}\left(\frac{\delta}{2}\right) \, , \, 1 - \left((m+1)\frac{\delta}{2} + \left(\frac{d}{|\mathbb{F}|}\right)^m \right) \right\} \ . \tag{5}$$

For any $2(m+1)$ provers $P^*_1, \ldots, P^*_{2(m+1)}$ the following holds:

$$\Pr\left[V^{P^*_1,\ldots,P^*_{2(m+1)}}_{\mathrm{VRM}}(\mathbb{F}, H, m, d) = 1 \right] \leq 1 - s^{\mathrm{VRM}}_{|\mathbb{F}|,|H|,m,d}\Big(\Delta\big(f_{P^*_1}, \mathrm{VRM}(\mathbb{F}, \mathbb{F}, H, m, d)\big) \Big) \ ,$$

where $f_{P^*_1} \colon \mathbb{F}^m \to \mathbb{F}$ is defined by $f_{P^*_1}(\vec{\alpha}) := P^*_1(\vec{\alpha})$ for every $\vec{\alpha} \in \mathbb{F}^m$.

Following the proof of [BSS08, Lemma 4.5]:

Proof. For $i = 0, \ldots, m$, define $f_{P^*_i} \colon \mathbb{F}^m \to \mathbb{F}$ by $f_{P^*_i}(\vec{\alpha}) := P^*_{i+1}(\vec{\alpha})$ for $\vec{\alpha} \in \mathbb{F}^m$. Define $\delta := \Delta\big(f_{P^*_0}, \mathrm{VRM}(\mathbb{F}, \mathbb{F}, H, d, m)\big)$.

By Lemma 6.3, if there exist $i \in \{0, 1, \ldots, m+1\}$ such that $f_{P^*_i}$ is $\frac{\delta}{2}$-far from $\mathrm{RM}(\mathbb{F}, \mathbb{F}, m, d)$, the verifier rejects with probability at least $s^{\mathrm{RM}}_{|\mathbb{F}|,m,d}(\frac{\delta}{2})$. So suppose that $f_{P^*_0}, \ldots, f_{P^*_m}$ are all $\frac{\delta}{2}$-close to $\mathrm{RM}(\mathbb{F}, \mathbb{F}, m, d)$.

In such a case, it must be that $f_{P^*_0}$ is $\frac{\delta}{2}$-close to a polynomial Q of degree at most d that does *not* vanish on H^m. Let $Q_1, \ldots, Q_m$ be the polynomials of degree at most d respectively closest to $f_{P^*_1}, \ldots, f_{P^*_m}$. Because of Lemma 6.7, we know that $Q(\vec{z}) \neq \sum_{i=1}^m Z_H(z_i) Q_i(\vec{z})$. Thus the two polynomials agree on at most d^m points. We deduce that the verifier accepts with probability at most $(m+1)\frac{\delta}{2} + d^m |\mathbb{F}|^{-m}$.

□

Remark 6.9 (number of provers)**.** Because the queries used for low-degree testing of each of the $(m+1)$ polynomials are also used for the consistency check, it is crucial that each of these queries goes to a different prover, and that is why we use $2 \cdot (m+1)$ provers.

Remark 6.10 (efficiency of V_{VRM})**.** The verifier V_{RM} only needs to sample three points, check for linear independence, evaluate $(m+1)$ trivariate polynomials of degree at most d, and then perform a consistency check. We have already noted in Remark 6.4 that all these operations, except the consistency check, can be done in $\mathrm{poly}(m+d)$ field operations. The consistency check involves computing the *vanishing polynomial* Z_H for the subset H; this can be done in $\tilde{O}(|H|)$ field operations. So, overall, V_{RM} only requires $\mathrm{poly}(m + d + |H|)$ field operations.[18]

Remark 6.11 (efficiency of P_{VRM})**.** Each honest prover pair $(P_{\mathrm{VRM}}|_i, P_{\mathrm{VRM}}|_{m+i})$ has the same efficiency as an honest prover pair for V_{RM}, with the additional computation necessary to figure out Q_i. By inspection of the proof of [BSS08, Lemma 4.9], this can also be done in time comparable to the running time of the honest prover pair. Overall, the prover's (time and space) complexity is dominated by having to evaluate Q at $\mathrm{poly}(|\mathbb{F}|)$ points.

[18] Unlike in [BSCGT13], H will eventually not be a large set, so that we do not need to ensure that Z_H can be evaluated in time $\mathrm{polylog}(|H|)$.

6.3 An MIP For Multivariate Polynomials

We assemble the MIP protocols for testing properties of polynomials constructed in Section 6.1 and Section 6.2 into a succinct MIP of knowledge for a certain succinct algebraic constraint satisfaction problem sACSP; in Sections 6.4 and 6.5, we will explain how to reduce to this problem with suitable efficiency.

The definition of sACSP is a modification of that in [Har04, Proposition 5.4.1]:

Definition 6.12. *The language* sACSP *is the set of all tuples* $(x, \mathbb{F}, H, m, D, I)$*, where*

- x *is a binary string,*
- $\mathbb{F}$ *is a finite field,*
- H *is a set such that* $\{0,1\} \subseteq H \subseteq \mathbb{F}$*,*
- $m \in \mathbb{N}$ *is a dimension,*
- $D\colon \mathbb{F}^{3m+3} \to \mathbb{F}$ *is a polynomial,*
- I *is a subset of* H *such that* $|I^m| = |x|$*,*
- $\frac{1}{4(m'+1)} \leq 3\varepsilon_{|\mathbb{F}|,m',d'} \leq \frac{1}{2(m'+1)}$ *where* $m' = 3m+3$ *and* $d' = \deg(D) + 3m|H|$ *(cf. Equation (2)),*

such that there exists a witness polynomial $A\colon \mathbb{F}^m \to \mathbb{F}$ *of degree at most* $m|H|$ *for which the polynomial*

$$K^{D,A}(z_1,\ldots,z_{3m+3}) := D(z_1,\ldots,z_{3m+3}) \cdot \big(A(z_1,\ldots,z_m) - z_{3m+1}\big) \cdot \big(A(z_{m+1},\ldots,z_{2m}) - z_{3m+2}\big) \cdot \big(A(z_{2m+1},\ldots,z_{3m}) - z_{3m+3}\big)$$

vanishes everywhere on H^m *and* $x = A|_{I^m}$

We construct a verifier V_{sACSP} and prover(s) P_{sACSP} for the language sACSP; we need

$$\ell := 2 + 2\big((3m+3)+1\big) + 2 + (2(m+1)-1) = 8m+13$$

provers. The construction of V_{sACSP} is as follows:

$V_{\text{sACSP}}^{P_1^*,\ldots,P_\ell^*}\big((x, \mathbb{F}, H, m, D, I)\big) \stackrel{\text{def}}{=}$

1. Compute parameters:
   (a) $m' := 3m+3$;
   (b) $d := m|H|$;
   (c) $d' := \deg(D) + 3m|H|$.
2. Draw randomness:
   (a) $\vec{\alpha}, \vec{\beta}_1, \vec{\beta}_2 \leftarrow \mathbb{F}^m$;
   (b) $\lambda_1, \lambda_2, \lambda_3 \leftarrow \mathbb{F}$;
   (c) $\vec{\beta'}_1, \vec{\beta'}_2 \leftarrow \mathbb{F}^{m'}$;
   (d) $\vec{\gamma}_1, \vec{\gamma}_2 \leftarrow \mathbb{F}^m$;
   (e) $b \leftarrow \{0,1\}$.
3. Ask all queries:
   (a) define $\vec{\alpha'} := (\vec{\alpha}, \vec{\beta}_1, \vec{\beta}_2, \lambda_1, \lambda_2, \lambda_3)$;
   (b) query P_1^* at $\vec{\alpha}$ to obtain ω;
   (c) query P_2^* at $(\vec{\alpha}, \vec{\beta}_1, \vec{\beta}_2)$ to obtain a trivariate polynomial Q of degree at most d;
   (d) query $P_3^*, \ldots, P_{2+(m'+1)}^*$ at $\vec{\alpha'}$ to obtain $\tau_0, \ldots, \tau_{m'}$; and
   (e) query $P_{2+(m'+1)+1}^*, \ldots, P_{2+2(m'+1)}^*$ at $(\vec{\alpha'}, \vec{\beta'}_1, \vec{\beta'}_2)$ to obtain trivariate polynomials $R_0, \ldots, R_{m'}$ of degree at most d';
   (f) if $b = 0$, query $P_{2+2(m'+1)+1}^*$ at $\vec{\alpha}$ and $P_{2+2(m'+1)+2}^*$ at $\vec{\alpha}$ to obtain ω_1 and ω_2;
   (g) if $b = 1$, query $P_{2+2(m'+1)+1}^*$ at $\vec{\gamma}_1$ and $P_{2+2(m'+1)+2}^*$ at $\vec{\gamma}_2$ to obtain ω_1 and ω_2;
   (h) query $P_{2+(m'+1)+2+1}^*, \ldots, P_{2+(m'+1)+2+m}^*$ at $\vec{\alpha}$ to obtain $\sigma_1, \ldots, \sigma_m$;

(i) query $P^*_{2+(m'+1)+2+m+1}, \ldots, P^*_{2+(m'+1)+2+m+(m+1)}$ at $(\vec{\alpha}, \vec{\beta}_1, \vec{\beta}_2)$ to obtain trivariate polynomials $S_0, \ldots, S_m$ of degree at most d.

4. If $\vec{\alpha}, \vec{\beta}_1, \vec{\beta}_2$ or $\vec{\alpha'}, \vec{\beta'}_1, \vec{\beta'}_2$ are linearly dependent, accept; otherwise continue.
5. If $b = 0$, verify that $\omega = \omega_1 = \omega_2$; otherwise continue.
6. Test assignment function for low degree:
   (a) verify that $\omega = Q(\vec{\alpha})$.
7. Test "composed polynomial" function for low degree and vanishing property:
   (a) compute $Z_H(z) = \prod_{\gamma \in H}(z - \gamma)$;
   (b) verify that $\tau_0 = R_0(\vec{\alpha'}), \ldots, \tau_{m'} = R_{m'}(\vec{\alpha'})$ and $\tau_0 = \sum_{i=1}^{m'} Z_H(\alpha'_i)\tau_i$.
8. Test consistency:
   (a) verify that $\tau_0 := D(\vec{\alpha}) \cdot (\omega - \lambda_1) \cdot (\omega_1 - \lambda_2) \cdot (\omega_2 - \lambda_3)$.
9. Test input:
   (a) Compute X to be the low-degree extension of x over the domain I^m;
   (b) Compute $Z_I(z) = \prod_{\gamma \in I}(z - \gamma)$;
   (c) Verify that $\tau_0 - X(\vec{\alpha}) = S_0(\vec{\alpha}), \sigma_1 = S_1(\vec{\alpha}), \ldots, \sigma_m = S_m(\vec{\alpha})$ and $\tau_0 - X(\vec{\alpha}) = \sum_{i=1}^{m} Z_I(\alpha_i)\sigma_i$.

The construction of P_{sACSP} is as follows:

$P_{\text{sACSP}}\big((x, \mathbb{F}, H, m, D, I), i, A; q\big) \stackrel{\text{def}}{=}$

1. Compute parameters as in Step (1) of V_{sACSP}.
2. If $i \in \{1, 2 + 2(m' + 1) + 1, 2 + 2(m' + 1) + 2\}$:
   (a) parse q as a query $\vec{\alpha} \in \mathbb{F}^m$;
   (b) compute and output $A(\vec{\alpha})$.
3. If $i = 2$:
   (a) parse q as a query $(\vec{\alpha}, \vec{\beta}_1, \vec{\beta}_2) \in \mathbb{F}^{3m}$;
   (b) compute and output the coefficients of the polynomial A when restricted to the subspace spanned by $(\vec{\alpha}, \vec{\beta}_1, \vec{\beta}_2)$.
4. If $i \in \{3, \ldots, 2 + (m' + 1)\}$:
   (a) parse q as a query $\vec{\alpha'} \in \mathbb{F}^{m'}$;
   (b) if $i = 3$, compute and output $K^{D,A}(\vec{\alpha'})$;
   (c) otherwise, compute and output $K^{D,A}_{i-3}(\vec{\alpha'})$.
5. If $i \in \{2 + (m' + 1) + 1, \ldots, 2 + 2(m' + 1)\}$:
   (a) parse q as a query $(\vec{\alpha'}, \vec{\beta'}_1, \vec{\beta'}_2) \in \mathbb{F}^{3m'}$;
   (b) if $i = 2 + (m' + 1) + 1$, compute and output the coefficients of the polynomial $K^{D,A}$ when restricted to the subspace spanned by $(\vec{\alpha'}, \vec{\beta'}_1, \vec{\beta'}_2)$;
   (c) otherwise, compute and output the coefficients of the polynomial $K^{D,A}_{i-2-(m'+1)-1}$ when restricted to the subspace spanned by $(\vec{\alpha'}, \vec{\beta'}_1, \vec{\beta'}_2)$.
6. If $i \in \{2 + 2(m' + 1) + 2 + 1, 2 + 2(m' + 1) + 2 + m\}$:
   (a) parse q as a query $\vec{\alpha} \in \mathbb{F}^m$;
   (b) compute X to be the low-degree extension of x over the domain I^m;
   (c) compute and output $(A - X)_{i-2-2(m'+1)-2}(\vec{\alpha})$.
7. If $i \in \{2 + 2(m' + 1) + 2 + m + 1, 2 + 2(m' + 1) + 2 + m + (m + 1)\}$:
   (a) parse q as a query $(\vec{\alpha}, \vec{\beta}_1, \vec{\beta}_2) \in \mathbb{F}^{3m}$;
   (b) if $i = 2 + 2(m' + 1) + 2 + m + 1$, compute and output the coefficients of the polynomial $(A - X)$ when restricted to the subspace spanned by $(\vec{\alpha}, \vec{\beta}_1, \vec{\beta}_2)$;
   (c) otherwise, compute and output the coefficients of the polynomial $(A - X)_{i-2-2(m'+1)-2-m-1}$ when restricted to the subspace spanned by $(\vec{\alpha}, \vec{\beta}_1, \vec{\beta}_2)$.

Before proceeding to prove the guarantees of V_{sACSP}, let us recall a well-known upper bound to the number of zeros of multivariate polynomials:

Lemma 6.13 (Schwartz–Zippel)**.** *Let $\mathbb{F}$ be a finite field and $Q\colon \mathbb{F}^m \to \mathbb{F}$ be a non-zero polynomial of degree at most d, with the maximum individual degree in each variable bounded by $|\mathbb{F}| - 1$. Then,*

$$\Pr_{\vec{\alpha} \leftarrow \mathbb{F}^m}\Big[Q(\vec{\alpha}) \neq 0\Big] \geq \frac{1}{|\mathbb{F}|^a} \cdot \left(1 - \frac{b}{|\mathbb{F}|}\right) \ ,$$

where $d = a \cdot (|\mathbb{F}| - 1) + b$ with $0 \leq b < |\mathbb{F}| - 1$.

We now prove that V_{sACSP} ensures a proof of knowledge property for the language sACSP:

Theorem 6.14. *There exists E such that, for any instance $(x, \mathbb{F}, H, m, D, I)$ and provers $P_1^*, \ldots, P_\ell^*$,*

$$\begin{aligned}\Pr\left[V_{\text{sACSP}}^{P_1^*,\ldots,P_\ell^*}\big((x, \mathbb{F}, H, m, D, I)\big) = 1\right] &> 1 - \frac{1}{100} \\ &\to \ \Big((x, \mathbb{F}, H, m, D, I), E^{P_1^*,\ldots,P_\ell^*}\big((x, \mathbb{F}, H, m, D, I)\big)\Big) \in \mathcal{R}(\text{sACSP}) \ .\end{aligned}$$

Proof. The definition of sACSP (cf. Definition 6.12) ensures that $3\varepsilon_{|\mathbb{F}|,m,d}, 3\varepsilon_{|\mathbb{F}|,m',d'} \leq \frac{1}{2(m'+1)}$. As before, let X be the low-degree extension of x over the domain I^m. Moreover:

- By Lemma 6.3,

$$\Pr\left[V_{\text{RM}}^{P_1^*,P_2^*}(\mathbb{F}, m, d) = 1\right] \leq 1 - \Delta\big(f_{P_1^*}, \text{RM}(\mathbb{F}, \mathbb{F}, m, d)\big) + \frac{1}{2(m'+1)} \ .$$

- By Lemma 6.8, for $\Delta\big(f_{P_3^*}, \text{VRM}(\mathbb{F}, \mathbb{F}, H, m', d')\big)\Big) \leq \frac{1}{m'+1}$,

$$\Pr\left[V_{\text{VRM}}^{P_3^*,\ldots,P_{2+2(m'+1)}^*}(\mathbb{F}, H, m', d') = 1\right] \leq 1 - \frac{\Delta\big(f_{P_3^*}, \text{VRM}(\mathbb{F}, \mathbb{F}, H, m', d')\big)}{2} + \frac{1}{2(m'+1)} \ .$$

- By Lemma 6.8, for $\Delta\big(f_{P_1^*} - X, \text{VRM}(\mathbb{F}, \mathbb{F}, I, m, d)\big)\Big) \leq \frac{1}{m'+1}$,

$$\Pr\left[V_{\text{VRM}}^{P_1^*,P_{2+2(m'+1)+2+1}^*,\ldots,P_{2+2(m'+1)+2+m+(m+1)}^*}(\mathbb{F}, I, m, d) = 1\right] \leq 1 - \frac{\Delta\big(f_{P_1^*} - X, \text{VRM}(\mathbb{F}, \mathbb{F}, I, m, d)\big)}{2} + \frac{1}{2(m'+1)} \ .$$

Now consider the following definitions:

- Define $\tilde{A}\colon \mathbb{F}^m \to \mathbb{F}$ by $\tilde{A}(\vec{\alpha}) := P_1^*(\vec{\alpha})$ for $\vec{\alpha} \in \mathbb{F}^m$.
- Define $\tilde{A}_1\colon \mathbb{F}^m \to \mathbb{F}$ by $\tilde{A}_1(\vec{\alpha}) := P_{2+2(m'+1)+1}^*(\vec{\alpha})$ for $\vec{\alpha} \in \mathbb{F}^m$.
- Define $\tilde{A}_2\colon \mathbb{F}^m \to \mathbb{F}$ by $\tilde{A}_2(\vec{\alpha}) := P_{2+2(m'+1)+2}^*(\vec{\alpha})$ for $\vec{\alpha} \in \mathbb{F}^m$.
- Define $\tilde{K}\colon \mathbb{F}^{m'} \to \mathbb{F}$ by $\tilde{K}(\vec{\alpha}) := P_3^*(\vec{\alpha})$ for $\vec{\alpha} \in \mathbb{F}^{m'}$.
- Define $\delta := \max\{\Delta(\tilde{A}, \tilde{A}_1), \Delta(\tilde{A}, \tilde{A}_2)\}$.
- Define $\delta_{\text{RM}} := \delta_{\text{VRM}} := \delta_x := \delta_c := \frac{1}{30}$.

We deduce that:

- $\delta \leq \delta_c$. If not, then the verifier would have rejected with probability at least $\frac{1}{2} \cdot \delta_c > \frac{1}{100}$, which is a contradiction.

- $\Delta(\tilde{A}, \mathrm{RM}(\mathbb{F}, \mathbb{F}, m, d)) < \delta_{\mathrm{RM}}$. If not, then the verifier would have rejected with probability greater than $\frac{1}{2} \cdot (\delta_{\mathrm{RM}} - \frac{1}{2(m'+1)}) > \frac{1}{100}$.

- $\Delta(\tilde{K}, \mathrm{VRM}(\mathbb{F}, \mathbb{F}, H, m, d')) < \delta_{\mathrm{VRM}}$. If not, then the verifier would have rejected with probability greater than $\frac{1}{2} \cdot (\delta_{\mathrm{VRM}}/2 - \frac{1}{2(m'+1)}) \geq \frac{1}{100}$, which is a contradiction.

- $\Delta(\tilde{A} - X, \mathrm{VRM}(\mathbb{F}, \mathbb{F}, I, m, d)) < \delta_x$. If not, then the verifier would have rejected with probability greater than $\frac{1}{2} \cdot (\delta_x/2 - \frac{1}{2(m'+1)}) \geq \frac{1}{100}$, which is a contradiction.

Let A be a polynomial whose evaluation lies in $\mathrm{RM}(\mathbb{F}, \mathbb{F}, m, d)$ that is closest to $\tilde{A}$. Note that $K^{D,A}$ has evaluation that lies in $\mathrm{RM}(\mathbb{F}, \mathbb{F}, m', d')$ and $K^{D,A}$ is $3\delta_{\mathrm{RM}}$-close to $K^{D,\tilde{A}}$. Let $\hat{K}$ be a polynomial whose evaluation lies in $\mathrm{VRM}(\mathbb{F}, \mathbb{F}, H, m', d')$ that is closest to $\tilde{K}$. We argue that that $K^{D,A}$ and $\hat{K}$ are equal. Indeed, suppose by way of contradiction that they are distinct; then, as they are both m'-variate polynomials of degree at most d', by a union bound and Lemma 6.13 (note that $d' < |\mathbb{F}| - 1$), the verifier would have accepted with probability at most $\frac{1}{2} + \frac{1}{2} \cdot (3\delta_{\mathrm{RM}} + \delta_{\mathrm{VRM}} + 2\delta_c + \frac{d'}{|\mathbb{F}|})$, i.e., the verifier would have rejected with probability greater than $\frac{1}{2} - \frac{1}{2} \cdot (3\delta_{\mathrm{RM}} + \delta_{\mathrm{VRM}} + 2\delta_c + \frac{d'}{|\mathbb{F}|}) \geq \frac{1}{100}$, which is a contradiction. Thus we know that A and $K^{D,A}$ have respective evaluations in $\mathrm{RM}(\mathbb{F}, \mathbb{F}, m, d)$ and $\mathrm{VRM}(\mathbb{F}, \mathbb{F}, H, m', d')$.

Next, note that $\tilde{A} - X$ is δ_{RM}-close to the evaluation table of $A - X$. Let A' be a polynomial of degree less than d vanishing on I^m whose evaluation table is closest to $\tilde{A} - X$. Note that $A' = A - X$ because both A' and $A - X$ are polynomials of degree less than d, the evaluation tables of A' and $A - X$ are $(\delta_{\mathrm{RM}} + \delta_x)$-close, and $1 - d/|\mathbb{F}| > \delta_{\mathrm{RM}} + \delta_x$. Hence, $A|_{I^m} = x$.

Thus, A is a "good" witness. We can simply define $E^{P_1^*, \ldots, P_\ell^*}\big((x, \mathbb{F}, H, m, D, I)\big)$ to be the algorithm that runs Reed–Muller decoding for the function $\tilde{A}$ induced by P_1^* to obtain the polynomial A. □

Remark 6.15 (efficiency of V_{sACSP})**.** We have already discussed the time complexity of V_{RM} and V_{VRM} in Remark 6.4 and Remark 6.10 respectively. We deduce that V_{sACSP} only requires $\mathrm{poly}(m|H|)$ field operations, plus however many field operations are required to evaluate D at a single point, plus $\tilde{O}(|x|)$ field operations to evaluate X at a single point.

Remark 6.16 (efficiency of P_{sACSP})**.** We have already discussed the time complexity of P_{RM} and P_{VRM} in Remark 6.5 and Remark 6.11 respectively. With these discussions in mind, it is easy to see that the (time and space) complexity of P_{sACSP} is dominated by $\mathrm{poly}(|\mathbb{F}|)$ evaluations of D, A, and X. (Note that any given evaluation of X requires $\tilde{O}(|x|)$ field operations.)

Remark 6.17 (query generation)**.** The generation of queries (see Step (2) and Step (3)) only depends on the "size" of the instance; specifically, the queries are essentially a random string of length $O(m \log |\mathbb{F}|)$ (and do not depend on, e.g., D). In particular, we can "split" V_{sACSP} into a probabilistic query generator G that only takes $(|\mathbb{F}|, m)$ as input, and a deterministic verification test V that takes as input the vector of queries $\vec{q}$, the provers' replies, and the full instance $(x, \mathbb{F}, H, m, D, I)$. The ability to perform this split is important to enable "adaptivity" in the succinct argument constructions of this paper.

Remark 6.18 (local witness decoding)**.** If one is interested in an extractor that achieves *local* decoding of the witness (cf. Remark 5.5), then one can simply let the extractor use any local decoding algorithm for Reed–Muller codes.

Remark 6.19 (efficient reverse sampler)**.** It is easy to see that V_{sACSP} has an *efficient reverse sampler* (when properly defined for MIPs by modifying the definition of [BG08]).

6.4 An MIP For Circuits

We construct an MIP protocol for (succinctly-described) boolean circuits. In light of the MIP protocol for sACSP from Section 6.3, we only need to show how to appropriately arithmetize a boolean circuit, i.e., reduce from succinct boolean circuit satisfiability to sACSP. For this part, we modify the ideas in [Har04, Proposition 5.4.1] to work for *succinctly-described* circuits. (Recall that the verifier does not have time to work in time proportional to the circuit size!)

Definition 6.20. *Let $n \in \mathbb{N}$. A boolean formula $\phi\colon \{1,\ldots,n\}^3 \times \{0,1\}^3 \to \{0,1\}$ describes a boolean circuit C of size n with wires $w_1,\ldots,w_n$ if, for all $i_1,i_2,i_3 \in \{1,\ldots,n\}$ and $b_1,b_2,b_3 \in \{0,1\}$,*

$$\phi(i_1,i_2,i_3,b_1,b_2,b_3) := \begin{cases} 1 & \text{there exists a gate in } C \text{ whose input variables and output variables} \\ & \text{are in the set } \{w_{i_1},w_{i_2},w_{i_3}\} \text{ and } w_{i_1}=\overline{b_1}, w_{i_2}=\overline{b_2}, w_{i_3}=\overline{b_3} \\ & \text{is an invalid configuration for this gate} \\ 0 & \text{otherwise} \end{cases}.$$

We refer to ϕ as a circuit descriptor.

Definition 6.21 (Succinct Circuit SAT)**.** *The language* sCSAT *is the language of all (x,ϕ) for which $C(x,a)=1$ for some assignment a, where C is the circuit described by ϕ.*

Lemma 6.22. *There is an efficient reduction that maps every (x,ϕ), where ϕ describes a circuit C of size n, to an instance (x,H,m,D,I) with $|H^m|=n$ and $m \le \frac{\log n}{\log\log n}$ such that $C(x,a)=1$ for some a if and only if $(x,\mathbb{F},H,m,D,I) \in$ sACSP. Furthermore, the reduction is "systematic" in the following sense:*

- Witness reduction. *If a is such that $C(x,a)=1$ then (x,a) can be extended to its corresponding evaluation $w\colon \{1,\ldots,n\} \to \{0,1\}$, which can then be low-degree-extended over H^m (cf. Definition 6.1) to a polynomial $A\colon \mathbb{F}^m \to \mathbb{F}$ that is a valid witness for $(x,\mathbb{F},H,m,D,I)$.*
- Inverse witness reduction. *If $A\colon \mathbb{F}^m \to \mathbb{F}$ is a polynomial that is a valid witness for $(x,\mathbb{F},H,m,D,I)$ then $w := A|_{H^m}$ is a valid and satisfying evaluation of $C(x,a)$ for some a.*

Proof sketch. Let $\mathbb{F}$ be a field, I,H sets such that $\{0,1\} \subseteq I \subseteq H \subseteq \mathbb{F}$, and $m \in \mathbb{N}$ such that $|I^m| = |x|$ and $|H^m| = n$; without loss of generality, $|H| = 2^r$ for some $r \in \mathbb{N}$. The size of $\mathbb{F}$ should be chosen to satisfy the condition in Definition 6.12. Our job now is to construct D out of ϕ.

Let $\Pi_i\colon \mathbb{F} \to \mathbb{F}$ be the polynomial that "projects" every element $\alpha \in H$ to its i-th bit (when α is represented as a bit string); note that $\deg(\Pi_i) \le |H|$. Let Φ be the "direct" arithmetization of ϕ (i.e., first turn ϕ to a boolean formula of only ANDs and NOTs and then replace each AND by field multiplication and each NOT by "1 minus z"); note that Φ is a polynomial with $3mr+3$ variables with total degree bounded by $|\phi|$. Define $Z\colon \mathbb{F} \to \mathbb{F}$ to be the polynomial that equals 1 on $\{0,1\}$ but is 0 everywhere else on H; note that the degree of Z is at most $|H|$.

Define the polynomial $D\colon \mathbb{F}^{3m+3} \to \mathbb{F}$:

$$D\Big((z_{1,j})_{j=1}^m,(z_{2,j})_{j=1}^m,(z_{3,j})_{j=1}^m,y_1,y_2,y_3\Big) := \Phi\left(\begin{array}{l} \big(\Pi_1(z_{1,j}),\ldots,\Pi_r(z_{1,j})\big)_{j=1}^m, \\ \big(\Pi_1(z_{2,j}),\ldots,\Pi_r(z_{2,j})\big)_{j=1}^m, \\ \big(\Pi_1(z_{3,j}),\ldots,\Pi_r(z_{3,j})\big)_{j=1}^m, \\ y_1,y_2,y_3 \end{array}\right) \cdot \prod_{i=1}^3 Z(y_i) \ .$$

When restricted to inputs in $H^{3m} \times \{0,1\}^3$, D computes ϕ; everywhere else in H^{3m+3}, D evaluates to 0. The total degree of D is at most $(|\phi|+3) \cdot |H|$. It is easy to see that D yields the desired reduction, with the appropriate witness properties. □

Given the (Levin) reduction from Lemma 6.22, we immediately obtain a succinct MIP of knowledge $(P_{\text{sCSAT}}, V_{\text{sCSAT}})$ for circuit satisfiability:

- $V_{\text{sCSAT}}\big((x,\phi)\big)$ invokes the reduction above to obtain an instance $(x, \mathbb{F}, H, m, D, I)$ and then invokes $V_{\text{sACSP}}\big((x, \mathbb{F}, H, m, D, I)\big)$;
- $P_{\text{sCSAT}}\big((x,\phi), a\big)$ invokes the reduction above to obtain an instance $(x, \mathbb{F}, H, m, D, I)$, evaluates $C(x,a)$ to obtain the wire values w and then low-degree extends w to obtain a polynomial A, and then invokes $P_{\text{sACSP}}\big((x, \mathbb{F}, H, m, D, I), A\big)$.

The security properties of $(P_{\text{sCSAT}}, V_{\text{sCSAT}})$ follow from those of $(P_{\text{sACSP}}, V_{\text{sACSP}})$ and the reduction from sCSAT to sACSP.

Remark 6.23 (efficiency of V_{sCSAT})**.** We know from Remark 6.15 that the running time of V_{sACSP} is dominated by $\text{poly}(m|H|) + \tilde{O}(|x|)$ field operations plus a single evaluation of D. The reduction from Lemma 6.22 ensures that $m, |H| = O(\log n)$; furthermore, D can be evaluated with $|\phi| + \text{poly}(|H|)$ field operations. Overall, the running time of V_{sCSAT} is $(|\phi| + |x|) \cdot \text{polylog}(n)$.

Remark 6.24 (efficiency of P_{sCSAT})**.** We know from Remark 6.16 that the (time and space) complexity of P_{sACSP} is dominated by $\text{poly}(|\mathbb{F}|)$ evaluations of D, A, and X (which is the low-degree extension of x on I^m). The reduction from Lemma 6.22 ensures that $|\mathbb{F}| = |\phi| \cdot \text{polylog}(n)$ and that A is the low-degree extension of w (the wire values of the circuit described by ϕ on input (x,a)). Thus, $\text{poly}(|\mathbb{F}|)$ evaluations of D can be done in $\text{poly}(|\phi|) \cdot \text{polylog}(n)$ time, $\text{poly}(|\mathbb{F}|)$ evaluations of A can be done in $\text{poly}(|\phi|) \cdot n \cdot \text{polylog}(n)$ time, and $\text{poly}(|\mathbb{F}|)$ evaluations of X can be done in $\text{poly}(|\phi|) \cdot |x| \cdot \text{polylog}(n)$ time. The space required for all of these evaluations is only $(|\phi| + |x|) \cdot \text{polylog}(n)$, when having oracle access to A. Overall, the time and space of P_{sCSAT} respectively are $\text{poly}(|\phi|) \cdot (|x| + n) \cdot \text{polylog}(n)$ and $(|\phi| + |x|) \cdot \text{polylog}(n)$, when having oracle access to w.

Remark 6.25 (query separation)**.** We know from Remark 6.17 that the queries of V_{sACSP} are specified by a random string of length $O(m \log |\mathbb{F}|)$. Thus, because $m = O(\log n)$ and $|\mathbb{F}| = \text{poly}(|\phi|) \cdot n \cdot \text{polylog}(n)$, the queries of V_{sCSAT} are specified by a random string of length $O(\log |\phi| + \log(n))$.

We now introduce the notion of a t-time s-space circuit C:

Definition 6.26. *A t-time s-space circuit C is a circuit for which there is a t-time s-space random-access machine M such that, for every assignment a, $M(a)$ writes in its memory all the wire values of C on input a. In particular, any set W of values can be obtained in time $t + |W|$ and space $\max\{s, W\}$.*

In light of Remark 6.23, Remark 6.24, and Remark 6.25, we deduce the following theorem:

Theorem 6.27. *There exists a succinct MIP system $(P_{\text{sCSAT}}, V_{\text{sCSAT}})$ of knowledge for* sCSAT *such that, for every (x,ϕ) where ϕ describes a boolean circuit C of size n such that $C(x,a) = 1$ for some a, the following conditions hold:*

- $V_{\text{sCSAT}}\big((x,\phi)\big)$ *runs in time* $(|\phi| + |x|) \cdot \text{polylog}(n)$*; its queries are a random string of* $O(\log |\phi| + \log(n))$ *bits.*

- $P_{\text{sCSAT}}\big((x,\phi),a\big)$ *runs in time* $\text{poly}(|\phi|)\cdot(|x|+n)\cdot\text{polylog}(n)$
- *When having oracle access to the wire values* $w\colon\{1,\dots,n\}\to\{0,1\}$ *of* C *on the input* (x,a), $P_{\text{sCSAT}}\big((x,\phi),a\big)$ *runs in space* $(|\phi|+|x|)\cdot\text{polylog}(n)$.

In particular, if ϕ *describes a* t*-time* s*-space circuit* C *(cf. Definition 6.26), then* $P_{\text{sCSAT}}(\phi,a)$ *can be made to run in time* $\text{poly}(|\phi|)\cdot(|x|+t)\cdot\text{polylog}(t)$ *and space* $(|\phi|+|x|+s)\cdot\text{polylog}(t)$.

Remark 6.28. Compared to the arithmetization techniques for graph-coloring problems of Ben-Sasson et al. [BSCGT13], the arithmetization from Lemma 6.22 of the algebraic description F_C of a circuit C is "easy": it only requires taking the low-degree extension of the algebraic description of a circuit. The reason is that in our setting we can feel free to use multivariate polynomials and also not be concerned with the eventual size of the field; by using the MIP model, we avoid writing down evaluations over the entire field. In contrast, in [BSCGT13], they must use univariate polynomials and seek a reduction for which the field size is as small as possible.

6.5 An MIP For Random-Access Machines

We are now ready to prove Theorem 1. In light of the MIP protocol from Section 6.4, we only need to obtain a sufficiently efficient reduction from random-access machine computations (i.e., the universal language $\mathcal{L}_{\mathcal{U}}$ defined in Section 5.2) to sCSAT.

We invoke the reduction of Ben-Sasson et al. [BSCGT13] from $\mathcal{L}_{\mathcal{U}}$ to sCSAT that, roughly, ensures that a t-time s-space random-access machine M can be reduced to a circuit descriptor ϕ of size $O(|M|)$ describing a circuit C of size $O(|M|\log s)\cdot t$ such that an input (x,w) that makes M accept can be mapped to an evaluation of C in time $O(|M|\log s)\cdot t$ and space $O(|M|\log s)\cdot s$ — thus C is a t'-time s'-space circuit (cf. Definition 6.26) for $t'=O(|M|\log s)\cdot t$ and $s'=O(|M|\log s)\cdot s$. Additionally, this mapping is systematic in that w is a certain substring of the generated evaluation of C and, conversely, any satisfying evaluation of C contains as a substring a valid witness for M (with both having the first part of the input fixed to x).

We immediately obtain a succinct MIP of knowledge $(P_{\mathcal{L}_{\mathcal{U}}},V_{\mathcal{L}_{\mathcal{U}}})$ for $\mathcal{L}_{\mathcal{U}}$ as follows:

- $V_{\mathcal{L}_{\mathcal{U}}}\big((M,x,t)\big)$ invokes the reduction to obtain a circuit descriptor ϕ and then invokes $V_{\text{sCSAT}}\big((x,\phi)\big)$;
- $P_{\mathcal{L}_{\mathcal{U}}}\big((M,x,t),w\big)$ invokes the reduction to obtain a circuit descriptor ϕ, transforms w to a satisfying assignment a for the circuit described by ϕ, and then invokes $P_{\text{sCSAT}}\big((x,\phi),a\big)$.

In light of Theorem 6.27, we can then prove the following theorem:

Theorem 6.29. *There exists a succinct MIP system* $(P_{\mathcal{L}_{\mathcal{U}}},V_{\mathcal{L}_{\mathcal{U}}})$ *of knowledge for* $\mathcal{L}_{\mathcal{U}}$ *such that, for every instance* $(M,x,t)\in\mathcal{L}_{\mathcal{U}}$ *with valid witness* w *and* M *having space complexity* s*, the following conditions hold:*

- $V_{\mathcal{L}_{\mathcal{U}}}\big((M,x,t)\big)$ *runs in time* $(|M|+|x|)\cdot\text{polylog}(t)$*; its queries are random strings of* $O(\log|M|+\log t)$ *bits.*
- $P_{\mathcal{L}_{\mathcal{U}}}\big((M,x,t),w\big)$ *runs in time* $\text{poly}(|M|)\cdot(|x|+t)\cdot\text{polylog}(t)$ *and space* $(|M|+|x|+s)\cdot\text{polylog}(t)$.
- *Each MIP prover can be computed "gate-by-gate" as a circuit by an evaluator algorithm in time* $\text{poly}(|M|)\cdot(|x|+t)\cdot\text{polylog}(t)$ *and space* $(|M|+|x|+s)\cdot\text{polylog}(t)$.

Theorem 6.29 gives efficiency properties that are still not as good as those claimed in Theorem 1: the dependence on $|M|$ is not as good. But this can be easily fixed as follows. We can fix a universal random-access machine U where both $|U|$ is on the order of $\mathrm{polylog}(T)$, and then simulate a given random-access machine M (computing on a given (x, w)) on U step by step. The overhead at each step is polynomial in the register sizes, which is only $O(\log T)$. Thus, overall, simulating M on U is only $\mathrm{polylog}(T)$ slower than running M directly. Combining this observation with Theorem 6.29 yields a complexity-preserving MIP as claimed in Theorem 1.

Remark 6.30. In light of Remark 6.18 and Remark 6.19 and the fact that the reductions we used from $\mathcal{L}_{\mathcal{U}}$ to SACSP preserve both local decoding of the witness and efficient reverse samplability, we deduce that the complexity-preserving MIP we have constructed also enjoys the properties of local decoding of the witness (cf. Remark 5.5) and reverse samplability. These properties are ultimately needed for the construction of complexity-preserving universal arguments (cf. Remark 3.5).

7 Succinct Multi-Function Commitments

We define and construct *succinct multi-function commitment* (SMFC) schemes, which are the main tool in our construction of succinct interactive arguments from MIPs, described in the next section (Section 8).

At high-level, our goal is to allow a sender to succinctly commit to a function $f\colon \mathsf{Dom} \to \mathsf{Range}$ in a manner that allows succinct decommitment to specific values. As already explained in Section 3.3, the traditional way of doing this is computing the truth table of the function and then using a Merkle tree to commit to this table. However, this solution may require that the sender performs work $t|\mathsf{Dom}|$ where t is the time required to evaluate the function at a given point and $|\mathsf{Dom}|$ is the size of its domain. (In particular, if $|\mathsf{Dom}|$ is super-polynomial this may be infeasible.)

We would like to construct a scheme that allows the sender to commit to a function without having to evaluate it as so many points. An instance of such a commitment was constructed by Ishai et. al. for linear functions [IKO07], using encryption with linear homomorphism. In this section, we construct such commitments for *general* functions, based on fully homomorphic encryption.

A succinct multi-function commitment (SMFC) scheme is a two-party protocol between a sender S and a receiver R that informally works as follows;

- First, in a commitment phase, these two parties interact with the purpose of letting S commit to ℓ functions $(f_1, \ldots, f_\ell) \in \mathcal{F}$, where $\mathcal{F}$ is some prescribed family; at the end of this phase each party possesses a private decommitment state.
- Later, in a decommitment phase, the receiver R wishes to learn the value of each function $(f_1, \ldots, f_\ell)$ on a certain tuple of queries $(q_1, \ldots, q_\ell)$ of his choice. We shall assume for simplicity that the functions are all above some common domain Dom_k.

We require that if both parties behave honestly, R indeed learns $f_i(q_i)$ for all $i \in [\ell]$. Moreover, we require that the running time of the receiver only depends on the input and output length of the functions (as well as the security parameter) and is essentially independent of time required to evaluate the functions. For the sender, we require that the scheme is "time and space preserving", in the sense that the time and space required in order to commit and decommit to each function are at most a fixed polynomial in the security parameter and the time to evaluate *once* the function. (A stronger requirement, which we shall also address, requires that this dependence is quasilinear up to factors in the security parameter.)

In addition, the protocol should have the following (computational) binding property: for any efficient malicious sender S^*, after the commitment phase, for each $i \in [\ell]$, there is a single function $\tilde{f}_i$, according to which S^* can open the i-th query (without getting rejected by the verifier). In particular, this means that the answer to the i-th query is *independent* of the rest of the queries. We stress that the binding property only assures that there is some committed function $\tilde{f}_i$, and does not guarantee that this function is any specific function or that it satisfies any special properties.

We do not require that the commitment is hiding (although, our protocol may be adapted to also ensure hiding).

We now provide a formal definition. Both the receiver $\mathsf{R} = (\mathsf{R}_\mathsf{C}, \mathsf{R}_\mathsf{D})$ and the sender $\mathsf{S} = (\mathsf{S}_\mathsf{C}, \mathsf{S}_\mathsf{D})$ consist of two sub-protocols for commitment and decommitment. In the commitment stage the sender S_C has input $\vec{f} = (f_1, \ldots, f_\ell)$ and may toss coins s_1; R_C has no input and tosses random coins r_1. After interacting, both parties output a state $(\mathsf{st}_\mathsf{S}, \mathsf{st}_\mathsf{R}) \leftarrow \langle \mathsf{S}_\mathsf{C}(\vec{f}; s_1), \mathsf{R}_\mathsf{C}(\perp; r_1) \rangle$. In the decommitment stage, S_D is given st_S and may toss fresh random coins s_1; R_D is given as input a vector of queries $\vec{q} = (q_1, \ldots, q_\ell)$, the state st_R and fresh coins r_2. After interacting, R_D outputs a vector of values $(\perp, \vec{v}) \leftarrow \langle \mathsf{S}_\mathsf{D}(\mathsf{st}_\mathsf{S}; s_2), \mathsf{R}_\mathsf{D}(\vec{q}; \mathsf{st}_\mathsf{R}; r_2) \rangle$, where some of the vector's values maybe set to $\perp$, indicating rejection.

Definition 7.1 (Succinct multi-function commitment)**.** *An interactive protocol between a receiver* $\mathsf{R} = (\mathsf{R}_\mathsf{C}, \mathsf{R}_\mathsf{D})$ *and a sender* $\mathsf{S} = (\mathsf{S}_\mathsf{C}, \mathsf{S}_\mathsf{D})$ *is a* **succinct multi-function commitment** *for a class of functions* $\mathcal{F}$ *and parameter* $\ell = \ell(k)$*, if it satisfies the following requirements:*

1. ***Correctness.***

   For any security parameter $k \in \mathbb{N}$*, any tuple of functions* $\vec{f} := (f_1, \ldots, f_\ell) \in \mathcal{F}$ *over a common domain* Dom_k*, and any tuple of queries* $\vec{q} := (q_1, \ldots, q_\ell) \in \mathsf{Dom}_k$*:*

   $$\Pr_{\substack{r_1, r_2 \\ s_1, s_2}} \left[\frac{(\mathsf{st}_\mathsf{S}, \mathsf{st}_\mathsf{R}) \leftarrow \langle \mathsf{S}_\mathsf{C}(\vec{f}; s_1), \mathsf{R}_\mathsf{C}(\perp; r_1) \rangle}{(\perp, \vec{v}) \leftarrow \langle \mathsf{S}_\mathsf{D}(\mathsf{st}_\mathsf{S}; s_2), \mathsf{R}_\mathsf{D}(\vec{q}; \mathsf{st}_\mathsf{R}; r_2) \rangle} \wedge \forall i \in [\ell] : v_i = f_i(q_i) \right] \geq 1 - \mathrm{negl}(k) \ ,$$

   where $(r_1, r_2), (s_1, s_2)$ *are the random coins used by* R *and* S *in each one of the phases sampled from* $\{0,1\}^{\mathrm{poly}(k)}$ *(and the line separates the commitment phase from the decommitment phase).*

2. ***Succinctness and complexity-preservation.***

   The receiver runs in time $\mathrm{poly}(k) \cdot \ell \cdot (\log |\mathsf{Dom}_k| + \log |\mathsf{Range}_k|)$.

   For any vector of functions $\vec{f} = (f_1, \ldots, f_\ell)$ *with* $f_i \in \mathcal{F}$ *and* f_i *can be evaluated in time* t_i*, the sender runs in time* $\mathrm{poly}(k, \vec{t})$.

   We say that the commitment is **complexity-preserving***, if the sender runs in time* $\ell \cdot t \cdot \mathrm{poly}(k)$ *and space* $\ell \cdot s \cdot \mathrm{poly}(k)$*, where* t *and* s *are the time and space required to compute each function* f_i*.*

3. ***Binding.***

   After the commitment phase, the sender cannot open any query to two different values (independently of the rest of the quries). That is, for any poly-size $\mathsf{S}^* = (\mathsf{S}^*_\mathsf{C}, \mathsf{S}^*_\mathsf{D})$*, all large enough* $k \in \mathbb{N}$*, and any two tuples* $\vec{q}, \vec{q}' \in \mathsf{Dom}_k^\ell$*:*

   $$\Pr_{\substack{r_1, s_1 \\ r_2, r_2', s_2, s_2'}} \left[\begin{array}{c} \frac{(\mathsf{st}_{\mathsf{S}^*}, \mathsf{st}_\mathsf{R}) \leftarrow \langle \mathsf{S}^*_\mathsf{C}(s_1), \mathsf{R}_\mathsf{C}(\perp; r_1) \rangle}{(\perp, \vec{v}) \leftarrow \langle \mathsf{S}^*_\mathsf{D}(\mathsf{st}_{\mathsf{S}^*}; s_2), \mathsf{R}_\mathsf{D}(\vec{q}; \mathsf{st}_\mathsf{R}; r_2) \rangle} \wedge \exists i \in [\ell] : \begin{array}{l} q_i = q_i' \\ v_i \neq \perp \\ v_i' \neq \perp \\ v_i \neq v_i' \end{array} \\ (\perp, \vec{v}') \leftarrow \langle \mathsf{S}^*_\mathsf{D}(\mathsf{st}_{\mathsf{S}^*}; s_2'), \mathsf{R}_\mathsf{D}(\vec{q}'; \mathsf{st}_\mathsf{R}; r_2') \rangle \end{array} \right] \leq \mathrm{negl}(k) \ ,$$

where (r_1, s_1) are the coins used by R and S^* in the commitment phase, and (r_2, s_2) and (r'_2, s'_2) are two independent sets of coins for the decommitment phase (and the line separates the commitment phase from the decommitment phase).

Remark 7.2. Since quantification is done over all non-uniform adversaries, a standard averaging argument implies it suffices to consider deterministic S^* (by fixing s_1, s_2, s'_2 to the coins that maximizes the adversary's success). In this case, we will consider a deterministic $\mathsf{S}^*_{\mathsf{C}}$ and deterministic $(\mathsf{S}^*_{\mathsf{D}}, \mathsf{S}^{*\prime}_{\mathsf{D}})$.

The committed function: For any poly-size sender S^* and commitment coins (r_1, s_1) for R and S^*, we denote by $\mathsf{Val}^{\mathsf{S}^*}_{r_1,s_1}$ the circuit that, given queries $\vec{q} = (q_1, \ldots, q_\ell) \in \mathsf{Dom}_k$, decommitment coins (r_2, s_2), and $i \in [\ell]$, emulates the corresponding commitment and decommitment phases and outputs the i-th decommitment value v_i output by the receiver R (which may be $\perp$). The following claim states that if binding holds, then with overwhelming probability the commitment phase determines some function according to which the sender must decommit. (The claim is analogous to the extraction claim in [IKO07] for linear commitments.)

Claim 7.3. *If (S, R) is binding as in Definition 7.1, then for any polysize S^*, polynomial p, all large enough $k \in \mathbb{N}$, and any distribution $\mathcal{Q}$ on tuples of $\ell = \ell(k)$ queries in Dom_k, there exist functions $(\tilde{f}_1, \ldots, \tilde{f}_\ell)$ such that:*

$$\Pr_{r_1,s_1}\left[\Pr_{r_2,s_2,\vec{q}}\left[\exists i \in [\ell] : \mathsf{Val}^{\mathsf{S}^*}_{r_1,s_1}(\vec{q}; r_2, s_2; i) \notin \left\{\tilde{f}_i(r_1, s_1; q_i), \perp\right\}\right] \geq \frac{1}{p(k)}\right] \leq \mathrm{negl}(k) \ ,$$

where (r_1, r_2) are random coins for R, (s_1, s_2) are random coins for S^*, and $\vec{q}$ is sampled from $\mathcal{Q}$.

Proof sketch. Fix any sequence of distributions $\{\mathcal{Q}_k\}$. For any $q \in \mathsf{Dom}_k$, and (r_1, s_1), let $y = y(i, q)$ be the range element that maximizes

$$P(r_1, s_1; q; y; i) = \Pr_{\substack{r_2,s_2 \\ \vec{q} \leftarrow \mathcal{Q}_k(q,i)}}\left[y = \mathsf{Val}^{\mathsf{S}^*}_{r_1,s_1}(\vec{q}; r_2, s_2; i)\right] \ ,$$

where $\mathcal{Q}_k(q, i)$ is the distribution $\mathcal{Q}_k$ conditioned on the i-th element being q. We define $\tilde{f}_i(r_1, s_1, q) := y$.

Assume towards contradiction that there exists a polynomial p, such that for infinitely many k:

$$\Pr_{r_1,s_1}\left[\Pr_{\substack{r_2,s_2 \\ \vec{q} \leftarrow \mathcal{Q}_k}}\left[\exists i \in [\ell] : \mathsf{Val}^{\mathsf{S}^*}_{r_1,s_1}(\vec{q}; r_2, s_2; i) \notin \left\{\tilde{f}_i(r_1, s_1; q_i), \perp\right\}\right] \geq \frac{1}{p(k)}\right] \geq \frac{1}{p(k)}$$

Then, by an averaging argument, there is an infinite sequence of $i \in [\ell(k)]$ and $q \in \mathsf{Dom}_k$ such that:

$$\Pr_{r_1,s_1}\left[\Pr_{\substack{r_2,s_2 \\ \vec{q} \leftarrow \mathcal{Q}_k(q,i)}}\left[\mathsf{Val}^{\mathsf{S}^*}_{r_1,s_1}(\vec{q}; r_2, s_2; i) \notin \left\{\tilde{f}_i(r_1, s_1; q), \perp\right\}\right] \geq \frac{1}{2\ell p^2(k)}\right] \geq \frac{1}{2\ell p^2(k)}$$

For any (r_1, s_1) as above, we consider two cases, according to whether $P(r_1, s_1; q; y; i) > \frac{1}{6\ell p^2(k)}$ or $P(r_1, s_1; q; y; i) \leq \frac{1}{6\ell p^2(k)}$. We claim that in both cases. :

$$\Pr_{\substack{r_2,s_2 \\ r'_2,s'_2 \\ \vec{q},\vec{q}' \leftarrow \mathcal{Q}_k(q,i)}}\left[\mathsf{Val}^{\mathsf{S}^*}_{r_1,s_1}(\vec{q}; r_2, s_2; i) \neq \mathsf{Val}^{\mathsf{S}^*}_{r_1,s_1}(\vec{q}'; r'_2, s'_2; i)\right] \geq \frac{1}{36\ell^2 p^4(k)}$$

Indeed, in the first case, one value is $y = \tilde{f}_i(r_1, s_2, q)$ with probability at least $\frac{1}{6\ell p^2(k)}$ and the other is not y (nor $\perp$) with probability $\frac{1}{2\ell p^2(k)}$. In the second case, we can separate the set of values $\{y' \notin \{y, \perp\}\}$ into two disjoint sets, each with mass at least $\frac{1}{6\ell p^2(k)}$, indeed the total volume of this set is $\frac{1}{2\ell p^2(k)}$, and any element has mass at most $\frac{1}{6\ell p^2(k)}$ (as y takes the maximum).

This implies:

$$\Pr_{\substack{r_1,s_1 \\ r_2,r_2',s_2,s_2' \\ \vec{q},\vec{q}' \leftarrow \mathcal{Q}_k(q,i)}} \left[\frac{\begin{array}{c}(\mathsf{st}_{\mathsf{S}^*}, \mathsf{st}_{\mathsf{R}}) \leftarrow \langle \mathsf{S}_{\mathsf{C}}^*(s_1), \mathsf{R}_{\mathsf{C}}(\perp; r_1)\rangle \\ (\perp, \vec{v}) \leftarrow \langle \mathsf{S}_{\mathsf{D}}^*(\mathsf{st}_{\mathsf{S}^*}; s_2), \mathsf{R}_{\mathsf{D}}(\vec{q}; \mathsf{st}_{\mathsf{R}}; r_2)\rangle \\ (\perp, \vec{v}') \leftarrow \langle \mathsf{S}_{\mathsf{D}}^*(\mathsf{st}_{\mathsf{S}^*}; s_2'), \mathsf{R}_{\mathsf{D}}(\vec{q}'; \mathsf{st}_{\mathsf{R}}; r_2')\rangle \end{array}}{} \wedge \begin{array}{c} q_i = q_i' = q \\ v_i \neq \perp \\ v_i' \neq \perp \\ v_i \neq v_i' \end{array} \right] \geq \frac{1}{\mathrm{poly}(\ell \cdot p(k))} \ ;$$

in particular, by an averaging argument we can fix $\vec{q}, \vec{q}'$ that violate the binding property of Definition 7.1. □

Succinct multi-function commitments from succinct single-function commitments. We show that for any polynomial function $\ell = \ell(k)$, we can construct SMFCs as defined above, given a **succinct (single) function commitment** (SFC) scheme, i.e. where $\ell := 1$. This is done by composing in parallel ℓ instances of the single-function scheme independently for each function (and hence preserves the number of rounds).

Proposition 7.4. *Let* (S, R) *be an SFC scheme (for single functions). Then the protocol* $(\mathsf{S}^{(\ell)}, \mathsf{R}^{(\ell)})$ *induced by executing in parallel* (S, R) *independently for each query, is SMFC (as in Definition 7.1).*

Proof sketch. The correctness and succinctness of the protocol follows directly from those of the underlying protocol. The binding property is based on the fact that an adversary for the composed protocol can be transformed to an adversary for the single-query protocol, by simply simulating the other coordinates independently.

Formally, assume towards contradiction that there is a malicious sender $\mathsf{S}^{(\ell)^*}$ that breaks the binding property when interacting with $\mathsf{R}^{(\ell)}$. Namely, there exists an infinite sequence of pairs $\vec{q}, \vec{q}'$ for which $\mathsf{S}^{(\ell)^*}$ breaks the binding property with noticeable probability $\varepsilon = \varepsilon(k)$. In particular, there exists a corresponding sequence of $i^* \in [\ell]$ such that binding is violated on the i^*-th coordinate with noticeable probability ε/ℓ. We construct a sender S^* that breaks the binding property of the single function scheme; S^* has $(\vec{q}, \vec{q}', i^*)$ hardwired as non-uniform advice.

In the commitment phase, $\mathsf{S}_{\mathsf{C}}^*$ emulates $\ell - 1$ independent executions of $\mathsf{R}_{\mathsf{C}}^1, \ldots \mathsf{R}_{\mathsf{C}}^{i^*-1}, \ldots, \mathsf{R}_{\mathsf{C}}^{i^*-1}, \ldots, \mathsf{R}_{\mathsf{C}}^{\ell}$ for $\mathsf{S}_{\mathsf{C}}^{(\ell)^*}$. The i^* execution is conducted with the external receiver $\mathsf{R}_{\mathsf{C}} := \mathsf{R}_{\mathsf{C}}^i$.

In the decommitment phase, $\mathsf{S}_{\mathsf{D}}^*$ flips a fair coin, and according to the result will choose whether to emulate the receivers with $\vec{q}$ or $\vec{q}'$. Again it emulates $\ell - 1$ independent executions of $\mathsf{R}_{\mathsf{D}}^1, \ldots \mathsf{R}_{\mathsf{D}}^{i^*-1}, \ldots, \mathsf{R}_{\mathsf{D}}^{i^*-1}, \ldots, \mathsf{R}_{\mathsf{D}}^{\ell}$ with corresponding inputs taken from $\vec{q}$ or $\vec{q}'$ according to the outcome of the coin toss. Once again the i^*-th execution will be conducted with the external receiver $\mathsf{R}_{\mathsf{D}} := \mathsf{R}_{\mathsf{D}}^i$.

It is left to note that, with probability $1/4$, $\mathsf{S}^{(\ell)^*}$ is experiencing the exact same distribution as in a standard interaction, and hence binding will be broken with probability at least $\varepsilon/4\ell$. □

7.1 Constructing Succinct Single (and Multi) Function Commitments From FHE

In this section, we construct SFCs for general functions based on fully-homomorphic encryption. Specifically, we construct a protocol (S, R) for a single function (i.e., $\ell := 1$), and then, by invoking Proposition 7.4, we obtain an SMFC, $(\mathsf{S}^{(\ell)}, \mathsf{R}^{(\ell)})$, for any polynomial $\ell = \ell(k)$ number of functions.

As explained in the introduction, the high-level idea of the scheme is to extend cut-and-choose techniques that were considered in the context of delegation by Chung et al. [CKV10]. There, in order to delegate a given function F, the delegator computes in a pre-processing stage an encryption ez of 0^s and the homomorphic evaluation of F on this ez, resulting in $\hat{\mathsf{ez}}$. Then, in the online stage, in order to delegate the computation of some $q \in \{0,1\}^s$, it sends an encryption eq of the value and an encryption of ez in a random order. The worker should then apply the deterministic homomorphic evaluation procedure of F on both ciphers. To verify, the delegator, given two evaluation $\tilde{\mathsf{ez}}$ and $\tilde{\mathsf{eq}}$, simply compares its own pre-processed $\hat{\mathsf{ez}}$ to $\tilde{\mathsf{ez}}$. Indeed, any worker cannot cheat with probability noticeably greater than $1/2$. To increase security, the scheme is then repeated in parallel.

In our case, there is no explicit function F (the sender may choose any function F to commit to) and we are only interested in establishing that there is *some* function F according to which the sender S answers. A first naïve idea is to let the sender compute on its own the homomorphic evaluation $\hat{\mathsf{ez}}$, and then challenge him with $\hat{\mathsf{ez}}$ and $\hat{\mathsf{eq}}$ in the decommitment phase. Of course, this cannot work as is, because the sender receives ez in the commitment phase in the clear; we thus add another layer of FHE during the commitment phase, and send him an encryption of ez and have him homomorphically compute $\hat{\mathsf{ez}}$.

For this to actually work, it is not enough to send a single cipher eq; indeed, the sender may simply apply some deterministic function F on each cipher, rather then evaluating it homomorphically. In this case, he will always be consistent on identical ciphers, while for most distinct ciphers $\mathsf{eq}, \mathsf{eq}'$, the resulting $F(\mathsf{eq}), F(\mathsf{eq}')$ are likely to decrypt to different values. To make the above idea go through, it is enough to send *two* (or more) ciphertexts $\mathsf{eq}_1, \mathsf{eq}_2$ together with ez, and on top of checking consistency on ez, we will check that the evaluations $\hat{\mathsf{eq}}_1, \hat{\mathsf{eq}}_2$ decrypt to the same thing.

The resulting scheme that we get is weakly binding (the sender may break the binding with some constant probability). Unlike in the delegation case (where there is an explicit function), repeating this scheme in parallel, does not seem to admit a simple security proof and encounters similar difficulties to those in general parallel repetition of argument systems. Indeed, to overcome these difficulties we need to invoke a general parallel repetition theorem such as the one of Haitner [Hai09] or the one of Chung and Liu [CL10] (we will eventually use the latter one).

Overall, we show:

Theorem 7.5. *Assuming the existence of fully-homomorphic encryption:*

- *There exists a four-message succinct multi-function commitment for all polynomial-time functions.*
- *There exists a complexity-preserving multi-function commitment for any $\vec{f}$ such that each function in $\vec{f}$ can be computed "gate-by-gate" as a circuit by an evaluator algorithm in time t and space s (and complexity-preservation holds with respect to t and s).*

Remark 7.6. For complexity-preservation to hold in Theorem 2, it is required that the FHE evaluator algorithm operates "locally" in a gate-by-gate fashion. Indeed all known FHE algorithms consist of two "local" evaluator algorithms, one for addition and one for multiplication.

We start by describing and proving the security of a weakly binding scheme and then move on to discuss how to augment in order to obtain strong binding via parallel repetition. The complexity-preservation of our final scheme is discussed in Section 3.3.

Protocol 2

Let k be the security parameter and let $\mathcal{F}_k$ be a set of functions mapping $\{0,1\}^{s(k)}$ to $\{0,1\}^{s'(k)}$. Let $t \geq 2$ be some constant.

Main ingredient.

- A fully homomorphic encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$. (Can be a symmetric key scheme, in which case the cipher may include any required evaluation key.)

The commitment phase.

The receiver $\mathsf{R_C}$:

- Samples two symmetric keys: $\mathsf{sk} \leftarrow \mathsf{Gen}(1^k)$ and $\tilde{\mathsf{sk}} \leftarrow \mathsf{Gen}(1^k)$.
- Samples a double encryption of 0^s under the keys $(\mathsf{sk}, \tilde{\mathsf{sk}})$: let $\mathsf{ez} \leftarrow \mathsf{Enc}_{\mathsf{sk}}(0^s)$ and let $a \leftarrow \mathsf{Enc}_{\tilde{\mathsf{sk}}}(\mathsf{ez})$.
- Sends a to $\mathsf{S_C}$, and keeps the state $\mathsf{st_R} = (\mathsf{sk}, \tilde{\mathsf{sk}}, \mathsf{ez})$.

The sender $\mathsf{S_C}$:

- Given input function $f \in \mathcal{F}$, let Eval_f denote the deterministic function that, given a cipher c of an s bit message, computes $\mathsf{Eval}(f, \mathsf{c})$.
- Given the challenge a, computes $c := \mathsf{Eval}(\mathsf{Eval}_f, a)$.
  (That is, homomorphically evaluates the homomorphic evaluation of f.).
- Send c to $\mathsf{R_C}$ as the commitment.

The decommitment phase.

The receiver $\mathsf{R_D}$:

- Given input query $q \in \{0,1\}^s$, samples t independent encryptions of q under the (inner) secret key sk, i.e. $(\mathsf{eq}_1, \ldots, \mathsf{eq}_t) \leftarrow \mathsf{Enc}_{\mathsf{sk}}(q)$.
- Samples a random permutation $\sigma\colon\ \{0, 1, \ldots, t\} \to \{0, 1, \ldots, t\}$.
- Sends $\bar{q} := \sigma(\mathsf{ez}, \mathsf{eq}_1, \ldots, \mathsf{eq}_t)$ to $\mathsf{S_D}$, and keeps σ.
  (We denote by $\sigma(u_0, u_1, \ldots, u_t)$ the permuted values $(u_{\sigma(0)}, u_{\sigma(1)}, \ldots, u_{\sigma(t)})$.)

The sender $\mathsf{S_D}$:

- Given the tuple $\bar{q} = (\mathsf{c}_0, \mathsf{c}_1, \ldots, \mathsf{c}_t)$, compute $d_i = \mathsf{Eval}(f, \mathsf{c}_i)$ for $i = 0, 1, \ldots, t$.
- Send $d = (d_0, d_1, \ldots, d_t)$ to $\mathsf{R_D}$.

The receiver $\mathsf{R_D}$ - verification and output:

- Given the commitment c, the decommitment $(d_0, d_1, \ldots, d_t)$, and $(\mathsf{sk}, \tilde{\mathsf{sk}}), \sigma$.
- Computes $\hat{\mathsf{ez}} = \mathsf{Dec}_{\tilde{\mathsf{sk}}}(c)$ and checks that $d_{\sigma(0)} = \hat{\mathsf{ez}}$.
- Checks that $\mathsf{Dec}_{\mathsf{sk}}(d_{\sigma(1)}) = \mathsf{Dec}_{\mathsf{sk}}(d_{\sigma(2)}) = \cdots = \mathsf{Dec}_{\mathsf{sk}}(d_{\sigma(t)})$.
- If one of these checks fails, output $v := \perp$; otherwise, output $v := \mathsf{Dec}_{\mathsf{sk}}(d_{\sigma(1)})$.

Figure 2: A weakly-binding SFC from FHE.

7.2 A 4-Message Weakly-Binding Protocol

Following the intuition described above, we construct a weakly binding scheme as detailed in Figure 2. We now show that Protocol 2 is weakly binding.

Proposition 7.7. *Any malicious sender fails to break the binding property of Protocol 2 with constant probability. That is, there exist a constant $\delta < 1$, such that for any (deterministic) polysize $\mathsf{S}^* = (\mathsf{S}^*_{\mathsf{C}}, \mathsf{S}^*_{\mathsf{D}}, \mathsf{S}^{*\prime}_{\mathsf{D}})$, any large enough security parameter $k \in \mathbb{N}$, and any $q \in \mathsf{Dom}_k$:*

$$\Pr\left[\begin{array}{c} \mathsf{ez} \leftarrow \mathsf{Enc}_{\mathsf{sk}}(0^s) \\ a \leftarrow \mathsf{Enc}_{\tilde{\mathsf{sk}}}(\mathsf{ez}) \\ \hline (\mathsf{eq}_1, \ldots, \mathsf{eq}_t) \leftarrow \mathsf{Enc}_{\mathsf{sk}}(q) \\ \bar{q} \leftarrow \sigma(\mathsf{ez}, \mathsf{eq}_1, \ldots, \mathsf{eq}_t) \\ (\mathsf{eq}'_1, \ldots, \mathsf{eq}'_c) \leftarrow \mathsf{Enc}_{\mathsf{sk}}(q) \\ \bar{q}' \leftarrow \sigma'(\mathsf{ez}, \mathsf{eq}'_1, \ldots, \mathsf{eq}'_t) \end{array} \wedge \begin{array}{c} c \leftarrow \mathsf{S}^*_{\mathsf{C}}(a) \\ \hline (d_0, d_1, \ldots, d_t) \leftarrow \mathsf{S}^*_{\mathsf{D}}(a, \bar{q}) \\ (d'_0, d'_1, \ldots, d'_t) \leftarrow \mathsf{S}^{*\prime}_{\mathsf{D}}(a, \bar{q}') \end{array} \wedge \begin{array}{c} \mathsf{Dec}_{\tilde{\mathsf{sk}}}(c) = d_{\sigma(0)} = d'_{\sigma'(0)} \\ \mathsf{Dec}_{\mathsf{sk}}(d_{\sigma(1)}) = \cdots = \mathsf{Dec}_{\mathsf{sk}}(d_{\sigma(t)}) \\ \mathsf{Dec}_{\mathsf{sk}}(d'_{\sigma'(1)}) = \cdots = \mathsf{Dec}_{\mathsf{sk}}(d'_{\sigma'(t)}) \\ \mathsf{Dec}_{\mathsf{sk}}(d_{\sigma(1)}) \neq \mathsf{Dec}_{\mathsf{sk}}(d_{\sigma'(1)}) \end{array}\right] \leq \delta \ ,$$

where the probability is taken over the receiver's coins for the commitment phase, and the two independent decommitment phases, including all the secret keys and encryption coins and the permutations σ, σ'. (Again, we abuse notation and denote by $\sigma(u_0, u_1, \ldots, u_t)$ the permuted values $(u_{\sigma(0)}, u_{\sigma(1)}, \ldots, u_{\sigma(t)})$.)

Moreover, δ decreases with t; specifically $\delta = O(1/\sqrt{t})$.

Proof sketch. The proof of the lemma is shown in two steps. First, we consider a variant of Protocol 7.7 in a "split model" where the view of any malicious sender in the decommitment phase is completely independent of its view in the commitment phase. We show, based on the semantic security of the outer layer FHE, that an adversary breaking the protocol in the original model can be transformed into an adversary that also breaks (a variant of) the protocol in the split model.

In the second step, invoking the semantic security of the inner layer of FHE, we will bound the success probability of any adversary in breaking binding in this split model and conclude security of the original protocol.

The split model. In this model a malicious sender $\bar{\mathsf{S}}^*$ consists of three (deterministic) polysize algorithms $(\overline{\mathsf{S}}^*_{\mathsf{C}}, \overline{\mathsf{S}}^*_{\mathsf{D}}, \overline{\mathsf{S}}^{*\prime}_{\mathsf{D}})$. In the commitment phase, the receiver $\overline{\mathsf{R}}_{\mathsf{C}}$ sends $\mathsf{ez} \leftarrow \mathsf{Enc}_{\mathsf{sk}}(0^s)$ to $\overline{\mathsf{S}}^*_{\mathsf{C}}$ in the clear (rather than encrypted under an outer secret key), and gets back an evaluation $\hat{\mathsf{ez}}$ that is allegedly $\mathsf{Eval}(f, \mathsf{ez})$ for some function f. The decommitment phase is done as in the original game, except that $(\overline{\mathsf{S}}^*_{\mathsf{D}}, \overline{\mathsf{S}}^{*\prime}_{\mathsf{D}})$ do not get any state from the commitment phase. That is, each of $(\overline{\mathsf{S}}^*_{\mathsf{D}}, \overline{\mathsf{S}}^{*\prime}_{\mathsf{D}})$ only obtains $t+1$ randomly permuted encryptions $\sigma(\mathsf{ez}, \mathsf{eq}_1, \ldots, \mathsf{eq}_t)$ and $\sigma'(\mathsf{ez}, \mathsf{eq}'_1, \ldots, \mathsf{eq}'_t)$ (correspondingly); each one of $(\overline{\mathsf{S}}^*_{\mathsf{D}}, \overline{\mathsf{S}}^{*\prime}_{\mathsf{D}})$ then decommits to $t+1$ evaluations $(d_0, d_1, \ldots, d_t)$ and $(d'_0, d'_1, \ldots, d'_t)$. Finally, the receiver checks consistency with $\overline{\mathsf{S}}^*_{\mathsf{D}}$, i.e. $\hat{\mathsf{ez}} = d_{\sigma(0)} = d'_{\sigma'(0)}$ and that $\mathsf{Dec}_{\mathsf{sk}}(d_{\sigma(1)}) = \cdots = \mathsf{Dec}_{\mathsf{sk}}(d_{\sigma(t)}) \neq \mathsf{Dec}_{\mathsf{sk}}(d'_{\sigma'(1)}) = \cdots = \mathsf{Dec}_{\mathsf{sk}}(d'_{\sigma'(t)})$.

Using the semantic security of outer layer of FHE used in the original Protocol 2, we show that any adversary S^* that violates Proposition 7.7, can be transformed into a new adversary $\bar{\mathsf{S}}^*$ that breaks binding in the above split model (with some related probability).

Proposition 7.8. *Let S^* be a polysize malicious sender that for infinitely many $k \in \mathbb{N}$ and $q \in \mathsf{Dom}_k$, breaks the binding of Protocol 2 with probability larger than δ (i.e. violates Proposition 7.7). Then, there exists an adversary $\bar{\mathsf{S}}^*$ that breaks the binding of the protocol in the split model with probability at least $\delta^2 - \mathrm{negl}(k)$.*

Proof sketch. Starting from $\mathsf{S}^* = (\mathsf{S}^*_{\mathsf{C}}, \mathsf{S}^*_{\mathsf{D}}, \mathsf{S}^{*\prime}_{\mathsf{D}})$ that breaks the binding of the original protocol with probability at least δ, we construct $\bar{\mathsf{S}}^* = (\overline{\mathsf{S}}^*_{\mathsf{C}}, \overline{\mathsf{S}}^*_{\mathsf{D}}, \overline{\mathsf{S}}^{*\prime}_{\mathsf{D}})$ as follows.

Hybrid 1. We first consider a hybrid game where the value $\hat{\mathsf{ez}}$ for the consistency check is not computed as usual by decrypting the commitment message, but rather by emulating another decommitment phase. Specifically, we first sample $\mathsf{ez} \leftarrow \mathsf{Enc}_{\mathsf{sk}}(0^s)$ and $a \leftarrow \mathsf{Enc}_{\tilde{\mathsf{sk}}}(\mathsf{ez})$; in addition, we sample $\tilde{\mathsf{eq}}_1, \ldots, \tilde{\mathsf{eq}}_t \leftarrow \mathsf{Enc}_{\mathsf{sk}}(q)$ and a random permutation $\tilde{\sigma}$ and let $\bar{q} = \tilde{\sigma}(\mathsf{ez}, \tilde{\mathsf{eq}}_1, \ldots, \tilde{\mathsf{eq}}_t)$. Next, we run $\mathsf{S}^*_{\mathsf{D}}(a, \bar{q})$ to obtain $\tilde{d}_0, \tilde{d}_1, \ldots, \tilde{d}_t$ (this step is equivalent to running the sender $\mathsf{S}^*_{\mathsf{C}}$ with a in the commitment phase and then $\mathsf{S}^*_{\mathsf{D}}$ with $(a, \bar{q})$ in the decommitment phase). We set $\tilde{\hat{\mathsf{ez}}} := \tilde{d}_{\tilde{\sigma}(0)}$ to be the consistency check value.

Next, we execute the decommitment just as in the original experiment; namely, we sample fresh $(\mathsf{eq}_1, \ldots, \mathsf{eq}_t, \ldots, \mathsf{eq}'_1, \ldots, \mathsf{eq}'_c) \leftarrow \mathsf{Enc}_{\mathsf{sk}}(q)$ and fresh permutations σ, σ' and run $\mathsf{S}^*_{\mathsf{D}}(a, \sigma(\mathsf{ez}, \ldots, \mathsf{eq}_1, \mathsf{eq}_t))$ and $\mathsf{S}^{*\prime}_{\mathsf{D}}(a, \sigma(\mathsf{ez}, \ldots, \mathsf{eq}'_1, \mathsf{eq}'_t))$. However, in this hybrid, the success is not measured as usual by decrypting the commitment c of $\mathsf{S}^*_{\mathsf{C}}$, but rather using the value $\tilde{\hat{\mathsf{ez}}}$ extracted in the first part of the experiment. Success is again defined as achieving consistency of $\tilde{\hat{\mathsf{ez}}}$ with $d_{\sigma(0)}$ and $d'_{\sigma'(0)}$, and by satisfying $\mathsf{Dec}_{\mathsf{sk}}(d_{\sigma(1)}) = \cdots = \mathsf{Dec}_{\mathsf{sk}}(d_{\sigma(t)}) \neq \mathsf{Dec}_{\mathsf{sk}}(d'_{\sigma'(1)}) = \cdots = \mathsf{Dec}_{\mathsf{sk}}(d'_{\sigma'(t)})$.

Claim 7.9. *In the above hybrid experiment, the adversary succeeds with probability at least δ^2.*

Proof. Let $a = \mathsf{Enc}_{\tilde{\mathsf{sk}}}(\mathsf{Enc}_{\mathsf{sk}}(0^s))$ be a fixed encryption of a zero-encryption under fixed secret keys $(\mathsf{sk}, \tilde{\mathsf{sk}})$, as produced in the commitment phase. We consider the conditional probability over the coins for the decommitment phase that $(\mathsf{S}^*_{\mathsf{D}}, \mathsf{S}^{*\prime}_{\mathsf{D}})$ manage to equivocate in the original experiment (where consistency is checked by decrypting the commitment c):

$$\delta_{a,\mathsf{sk},\tilde{\mathsf{sk}}} := \Pr\left[\begin{array}{c} (\mathsf{eq}_1, \ldots, \mathsf{eq}_t) \leftarrow \mathsf{Enc}_{\mathsf{sk}}(q) \\ \bar{q} \leftarrow \sigma(\mathsf{ez}, \mathsf{eq}_1, \ldots, \mathsf{eq}_t) \\ (\mathsf{eq}'_1, \ldots, \mathsf{eq}'_c) \leftarrow \mathsf{Enc}_{\mathsf{sk}}(q) \\ \bar{q}' \leftarrow \sigma'(\mathsf{ez}, \mathsf{eq}'_1, \ldots, \mathsf{eq}'_t) \end{array} \wedge \frac{\begin{array}{c} c \leftarrow \mathsf{S}^*_{\mathsf{C}}(a) \end{array}}{\begin{array}{c} (d_0, d_1, \ldots, d_t) \leftarrow \mathsf{S}^*_{\mathsf{D}}(a, \bar{q}) \\ (d'_0, d'_1, \ldots, d'_t) \leftarrow \mathsf{S}^{*\prime}_{\mathsf{D}}(a, \bar{q}') \end{array}} \wedge \begin{array}{c} \mathsf{Dec}_{\tilde{\mathsf{sk}}}(c) = d_{\sigma(0)} = d'_{\sigma'(0)} \\ \mathsf{Dec}_{\mathsf{sk}}(d_{\sigma(1)}) = \cdots = \mathsf{Dec}_{\mathsf{sk}}(d_{\sigma(t)}) \\ \mathsf{Dec}_{\mathsf{sk}}(d'_{\sigma'(1)}) = \cdots = \mathsf{Dec}_{\mathsf{sk}}(d'_{\sigma'(t)}) \\ \mathsf{Dec}_{\mathsf{sk}}(d_{\sigma(1)}) \neq \mathsf{Dec}_{\mathsf{sk}}(d'_{\sigma'(1)}) \end{array}\right],$$

where the probability is taken over the receiver's coins in the two independent decommitment phases, including the encryption coins and the permutations σ, σ'.

First note that, since S^* wins the original game with probability at least δ, it holds that $\mathbb{E}\left[\delta_{a,\mathsf{sk},\tilde{\mathsf{sk}}}\right] \geq \delta$. Also, for any fixed $(a, \mathsf{sk}, \tilde{\mathsf{sk}})$, the equivocation probability $\delta_{a,\mathsf{sk},\tilde{\mathsf{sk}}}$ bounds from below the probability that $\mathsf{S}^*_{\mathsf{D}}$ satisfies the consistency check; namely, it holds that:

$$\Pr\left[\frac{\begin{array}{c} c \leftarrow \mathsf{S}^*_{\mathsf{C}}(a) \end{array}}{\begin{array}{c} (\mathsf{eq}_1, \ldots, \mathsf{eq}_t) \leftarrow \mathsf{Enc}_{\mathsf{sk}}(q) \\ \bar{q} \leftarrow \sigma(\mathsf{ez}, \mathsf{eq}_1, \ldots, \mathsf{eq}_t) \\ (d_0, d_1, \ldots, d_t) \leftarrow \mathsf{S}^*_{\mathsf{D}}(a, \bar{q}) \end{array}} \wedge d_{\sigma(0)} = \mathsf{Dec}_{\tilde{\mathsf{sk}}}(c)\right] \geq \delta_{(a,\mathsf{sk},\tilde{\mathsf{sk}})} \ .$$

where the probability is taken over the encryption coins and the permutation σ.

Moreover, conditioned on any $(a, \mathsf{sk}, \tilde{\mathsf{sk}})$, the probability that the sender wins in the new experiment (where consistency is checked with respect to $\tilde{\hat{\mathsf{ez}}}$, rather than with respect to the decryption of the commitment c) is at least as large as the probability that $(\mathsf{S}^*_{\mathsf{D}}, \mathsf{S}^{*\prime}_{\mathsf{D}})$ manage to equivocate in the original game and the value $\tilde{\hat{\mathsf{ez}}}$, extracted in the independent emulation of $\mathsf{S}^{*\,1}_{\mathsf{D}}$, was consistent with the decryption of c. Indeed, this exactly implies equivocation in the new experiment.

We can now get the required lower bound on the equivocation probability by averaging over $(a, \mathsf{sk}, \tilde{\mathsf{sk}})$:

$$\mathbb{E}_{(a,\mathsf{sk},\tilde{\mathsf{sk}})}\left[(\delta_{(a,\mathsf{sk},\tilde{\mathsf{sk}})})^2\right] \geq \left(\mathbb{E}_{(a,\mathsf{sk},\tilde{\mathsf{sk}})}\left[\delta_{(a,\mathsf{sk},\tilde{\mathsf{sk}})}\right]\right)^2 \geq \delta^2 \ .$$

□

Hybrid 2. We change the experiment given by hybrid 1 so that, instead of sampling the challenge a as an encryption of ez, it is sampled as an encryption of $\perp^{|\mathsf{ez}|}$.

Claim 7.10. *In this hybrid experiment, the adversary succeeds with probability at least* $\delta^2 - \mathrm{negl}(k)$.

Proof sketch. Indeed, in the previous hybrid experiment, we can test success without being given the outer secret key $\tilde{\mathsf{sk}}$ for the outer encryption (all that we need is the inner secret key). Hence, the result follows from the semantic security of the outer encryption. □

Obtaining $\bar{\mathsf{S}}^*$. We can now finally obtain $\bar{\mathsf{S}}^* = (\overline{\mathsf{S}}^*_{\mathsf{C}}, \overline{\mathsf{S}}^*_{\mathsf{D}}, \overline{\mathsf{S}}^{*\prime}_{\mathsf{D}})$ as follows. First, we may fix the secret key $\tilde{\mathsf{sk}}$ and encryption coins of the outer encryption used in the previous hybrid; also, we fix all the additional coins used to sample $\tilde{\mathsf{ez}}$ (as usual, the coins are fixed to maintain the success probability). Now, $\overline{\mathsf{S}}^*_{\mathsf{C}}$ is set to be the deterministic circuit that accepts ez as input and computes $\tilde{\mathsf{ez}}$ as prescribed by the first phase of the last hybrid. $(\overline{\mathsf{S}}^*_{\mathsf{D}}, \overline{\mathsf{S}}^{*\prime}_{\mathsf{D}})$ are set to be $(\mathsf{S}^*_{\mathsf{D}}, \mathsf{S}^{*\prime}_{\mathsf{D}})$ where the message a is a fixed encryption of $\perp^{|\hat{\mathsf{ez}}|}$ (using the already fixed coins for the outer encryption).

All and all we have obtained a split model adversary that (for infinitely many $k \in \mathbb{N}$) wins the split model game with probability at least $\delta^2 - \mathrm{negl}(k)$, which concludes the proof of Proposition 7.8.

□

Proposition 7.11. *Let* $\bar{\mathsf{S}}^* = (\overline{\mathsf{S}}^*_{\mathsf{C}}, \overline{\mathsf{S}}^*_{\mathsf{D}}, \overline{\mathsf{S}}^{*\prime}_{\mathsf{D}})$ *be any poly-size sender for the split model protocol. Then, for all large enough* $k \in \mathbb{N}$ *and* $q \in \mathsf{Dom}_k$, $\bar{\mathsf{S}}^*$ *breaks the binding property with probability at most* $\frac{32}{7(t+1)} + \frac{2^{11}}{t} + \mathrm{negl}(k)$.

Proof sketch. The proof is shown based on the semantic security of the inner encryption. Intuitively speaking, if $\bar{\mathsf{S}}^*$ manages to win the split model game, then he must be able to distinguish the 0^s-encryption ez from the two q-encryptions. Formally, we consider a semantic security game that corresponds to our split model.

The two-room semantic security game. The game involves a challenger and two adversaries $\mathcal{A}, \mathcal{A}'$. The challenger samples $2t+1$ ciphers $\mathsf{ez} \leftarrow \mathsf{Enc}_{\mathsf{sk}}(0^s)$ and $(\mathsf{eq}_1, \ldots, \mathsf{eq}_t, \mathsf{eq}'_1, \ldots, \mathsf{eq}'_t) \leftarrow \mathsf{Enc}_{\mathsf{sk}}(q)$. In addition, it samples two random permutations σ, σ' and hands $\sigma(\mathsf{ez}, \mathsf{eq}_1, \ldots, \mathsf{eq}_t)$ to $\mathcal{A}$ and $\sigma'(\mathsf{ez}, \mathsf{eq}'_1, \ldots, \mathsf{eq}'_t)$ to $\mathcal{A}'$. Each adversary then tries to guess which one of the ciphers is ez. The two adversaries cannot communicate (but rather only agree ahead of time on a strategy).

We denote by b the indicator for the event that $\mathcal{A}$ succeeds, and by b' the indicator for the event that $\mathcal{A}'$ succeeds. The goal of $\mathcal{A}, \mathcal{A}'$ is to maximize the expected value $\mathbb{E}[b_1 + b_2]$.

We first note that by semantic security it follows directly that, for any two poly size $\mathcal{A}, \mathcal{A}'$ (and all large enough $k \in \mathbb{N}$), it holds that $\mathbb{E}[b_1 + b_2] = \mathbb{E}[b_1] + \mathbb{E}[b_2] \leq \frac{2}{t+1} + \mathrm{negl}(k)$.

We now show that an adversary $\bar{\mathsf{S}}^* = (\overline{\mathsf{S}}^*_{\mathsf{C}}, \overline{\mathsf{S}}^*_{\mathsf{D}}, \overline{\mathsf{S}}^{*\prime}_{\mathsf{D}})$ that wins the split model game with probability ϵ, can be transformed into an adversary $(\mathcal{A}, \mathcal{A}')$ that in the semantic security two-room game obtains $\mathbb{E}[b+b'] \geq \frac{7}{16}\left(\epsilon - \frac{2^{11}}{t}\right)$. (Like $\bar{\mathsf{S}}^*$ the pair $(\mathcal{A}, \mathcal{A}')$ are deterministic.)

This is enough to conclude the proof of Proposition 7.11 as it shows that for all but finitely many $k \in \mathbb{N}$ $\epsilon(k) \leq \frac{32}{7(t+1)} + \frac{2^{11}}{t} + \mathrm{negl}(k)$.

The two-room adversary. Adversary $\mathcal{A}$ will be applying algorithms $\overline{\mathsf{S}}^*_{\mathsf{C}}$ and $\overline{\mathsf{S}}^*_{\mathsf{D}}$. Given a challenge $(\mathsf{c}_0, \mathsf{c}_1, \ldots, \mathsf{c}_t) = \sigma(\mathsf{ez}, \mathsf{eq}_1, \mathsf{eq}_2)$, $\mathcal{A}$ computes $(\hat{\mathsf{c}}_0, \hat{\mathsf{c}}_1, \ldots, \hat{\mathsf{c}}_t) \leftarrow \overline{\mathsf{S}}^*_{\mathsf{D}}(\mathsf{c}_0, \mathsf{c}_1, \ldots, \mathsf{c}_t)$. Then, for each $i \in \{0, 1, \ldots, t\}$, it compares $\overline{\mathsf{S}}^*_{\mathsf{C}}(\mathsf{c}_i)$ to $\hat{\mathsf{c}}_i$, and outputs a random i among the indices for which equality holds (if no equality holds it also outputs a random i).

Adversary $\mathcal{A}'$ is defined similarly, only that it invokes $\overline{\mathsf{S}}^{*\prime}_{\mathsf{D}}$ instead of $\overline{\mathsf{S}}^{*}_{\mathsf{D}}$. We shall denote its challenge by $(\mathsf{c}'_0, \mathsf{c}'_1, \ldots, \mathsf{c}'_t) = \sigma'(\mathsf{ez}, \mathsf{eq}'_1, \ldots, \mathsf{eq}'_t)$ and the output of $\overline{\mathsf{S}}^{*\prime}_{\mathsf{D}}$ by $(\hat{\mathsf{c}}'_0, \hat{\mathsf{c}}'_1, \ldots, \hat{\mathsf{c}}'_t)$.

We now analyze $\mathbb{E}[b_1+b_2]$. At high-level we show that, with high probability over the choice of $(\mathsf{ez}, \mathsf{eq}_1, \ldots, \mathsf{eq}_t, \mathsf{eq}'_1, \ldots, \mathsf{eq}'_t)$, and (σ, σ'), it holds that each $\mathsf{c}_{\sigma(0)}, \mathsf{c}'_{\sigma'(0)}$ (which is ez) is mapped by $\overline{\mathsf{S}}^{*}_{\mathsf{C}}$ to $\hat{\mathsf{c}}_{\sigma(0)}, \hat{\mathsf{c}}'_{\sigma'(0)}$, while at least one of $(\mathsf{c}_{\sigma(1)}, \ldots, \mathsf{c}_{\sigma(t)}, \mathsf{c}'_{\sigma'(1)}, \ldots, \mathsf{c}'_{\sigma'(t)})$ (which are $(\mathsf{eq}_1, \ldots, \mathsf{eq}_t, \mathsf{eq}'_1, \ldots, \mathsf{eq}'_t)$) is not mapped by $\overline{\mathsf{S}}^{*}_{\mathsf{C}}$ to the corresponding evaluation (among $(\hat{\mathsf{c}}_{\sigma(1)}, \ldots, \hat{\mathsf{c}}_{\sigma(t)}, \hat{\mathsf{c}}'_{\sigma'(1)}, \ldots, \hat{\mathsf{c}}'_{\sigma'(t)})$).

To do so, we first prove the following combinatorial claim:

Claim 7.12. *Let D be any distribution on functions with some common domain and let X be a distribution on this common domain. Consider the event E, over sampling $f \leftarrow D$ and a set Q of $2t$ elements $Q = R \cup R' = (x_1, \ldots, x_t, x'_1, \ldots, x'_t) \leftarrow X^{2t}$, that there exist sets $S = \{i_1, \ldots, i_m\} \subset \{1, \ldots, t\}$ and $T = \{i'_1, \ldots, i'_m\} \subset \{1, \ldots, t\}$ such that:*

$$m = |S| = |T| = \frac{3}{4}t \tag{6}$$

$$f(x_{i_1}) = \cdots = f(x_{i_m}) \neq f(x'_{i'_1}) = \cdots = f(x'_{i'_m}) \ , \tag{7}$$

then it holds that:

$$\Pr[E] \leq \frac{2^{11}}{t} \ .$$

Proof sketch. We will show that the above occurs for any function f in the support of D; thus, for the rest of the discussion fix some arbitrary f. Since the $2t$ elements are taken from the exact same distribution, we can think of first sampling the entire set Q and then choosing at random a subset $R \subseteq Q$ of size t to set as $(x_1, \ldots, x_t)$ (and the complement $R' = Q \setminus R$ as $(x'_1, \ldots, x'_t)$). It is enough to analyze the above probability conditioned on the event that there exist two images $y_1 \neq y_2$, such that for both $i \in \{1, 2\}$: $|f^{-1}(y_i) \cap Q| \geq \frac{3}{4}t$; namely, f "significantly-splits" the set Q. Let us denote by $Q_{f,i}$ the set $f^{-1}(y_i) \subseteq Q$. Note that, for the event E to occur, it must be that for some $i \in \{1, 2\}$ it holds that $\frac{3}{4}t \leq |Q_{f,i} \cap R| \leq \frac{5}{4}t$. However, we can use a tail bound to show that this occurs with small probability. Specifically, for each $x \in Q_{f,i}$, let I_x denote the indicator for the event $x \in R$, and let $x^* \neq y^*$ be to any two fixed elements in $Q_{f,i}$, then:

$$\mathbb{E}_R\left[|Q_{f,i} \cap R|\right] = \sum_{x \in Q_{f,i}} \mathbb{E}_R[I_x] = \frac{|Q_{f,i}|}{2} \leq \frac{1}{2} \cdot \frac{5t}{4} = (\frac{3}{4} - \frac{1}{8})t$$

$$\mathbb{V}\left[\sum_{x \in Q_{f,i}} I_x\right] = \sum_{(x,y) \in Q_{f,i} \times Q_{f,i}} (\mathbb{E}[I_x \cdot I_y] - \mathbb{E}[I_x]\mathbb{E}[I_y]) =$$

$$|Q_{f,i}|\left(\mathbb{E}[I_{x^*}^2] - (\mathbb{E}[I_{x^*}])^2\right) - \left(|Q_{f,i}|^2 - |Q_{f,i}|\right)\left(\mathbb{E}[I_{x^*} \cdot I_{y^*}] - (\mathbb{E}[I_{x^*}])^2\right) =$$

$$\frac{|Q_{f,i}|}{4} - \left(|Q_{f,i}|^2 - |Q_{f,i}|\right)\left(\frac{1}{2} \cdot \frac{t-1}{2t-1} - \frac{1}{4}\right) =$$

$$\frac{2|Q_{f,i}|(t-1) + |Q_{f,i}|^2}{4(2t-1)} \leq \frac{45t}{4} \ .$$

Hence, using a Chebyshev bound and a union bound on both $i \in \{1, 2\}$, it follows that $\Pr[E] \leq \frac{2^{11}}{t}$ as required. □

We now apply the above claim to the function distribution $\left\{\mathsf{Dec}_{\mathsf{sk}}(\overline{\mathsf{S}}^*_{\mathsf{C}}(\cdot)) : \mathsf{sk} \leftarrow \mathsf{Gen}(1^k)\right\}$, and for the input distribution $(\mathsf{eq}_1, \ldots, \mathsf{eq}_t, \mathsf{eq}'_1, \ldots, \mathsf{eq}'_t) \leftarrow \mathsf{Enc}_{\mathsf{sk}}(q)$ to obtain:

Corollary 7.13. *Let* G *be the event, over the choice of* $\mathsf{ciphers} := (\mathsf{ez}, \mathsf{eq}_1, \ldots, \mathsf{eq}_t, \mathsf{eq}'_1, \ldots, \mathsf{eq}'_t)$ *and* (σ, σ')*, that in the two-room game:*

1. $\overline{\mathsf{S}}^*_{\mathsf{C}}(\mathsf{c}_{\sigma(0)}) = \hat{\mathsf{c}}_{\sigma(0)} = \overline{\mathsf{S}}^*_{\mathsf{C}}(\mathsf{c}'_{\sigma'(0)}) = \hat{\mathsf{c}}'_{\sigma'(0)} = \mathsf{ez}$
2. *For* $\ell = t/4$*, there exists* $i_1, \ldots, i_\ell \in \{1, \ldots, t\}$ *such that, for each* $j \in \{1, \ldots, \ell\}$*, either* $\overline{\mathsf{S}}^*_{\mathsf{C}}(\mathsf{c}_{\sigma(i_j)}) \neq \hat{\mathsf{c}}_{\sigma(i_j)}$ *or* $\overline{\mathsf{S}}^*_{\mathsf{C}}(\mathsf{c}'_{\sigma'(i_j)}) \neq \hat{\mathsf{c}}'_{\sigma'(i_j)}$.

where $\{\mathsf{c}_i, \mathsf{c}'_i, \hat{\mathsf{c}}_i, \hat{\mathsf{c}}'_i\}_{i \in \{0,1,\ldots,t\}}$ *are the permuted* $\mathsf{ciphers}$ *as defined in the two-room game, then it holds that:*

$$\Pr[\mathsf{G}] \geq \epsilon - \frac{2^{11}}{t} \ .$$

Proof sketch. First note that, whenever $\bar{\mathsf{S}}^*$ wins the two-room game, the first condition of G holds and it also holds that:

$$\mathsf{Dec}_{\mathsf{sk}}\hat{\mathsf{c}}_{\sigma(1)} = \cdots = \mathsf{Dec}_{\mathsf{sk}}\hat{\mathsf{c}}_{\sigma(t)} \neq \mathsf{Dec}_{\mathsf{sk}}\hat{\mathsf{c}}'_{\sigma'(1)} = \cdots = \mathsf{Dec}_{\mathsf{sk}}\hat{\mathsf{c}}'_{\sigma'(t)}$$

Moreover, if in addition the second condition of G does **not** hold, then there exist two sets $S = \{i_1, \ldots, i_m\}$ and $T = \{i'_1, \ldots, i'_m\}$ of size $m = 3t/4$ such that:

$$\mathsf{Dec}_{\mathsf{sk}}(\overline{\mathsf{S}}^*_{\mathsf{C}}(\mathsf{c}_{\sigma(i_1)})) = \cdots = \mathsf{Dec}_{\mathsf{sk}}(\overline{\mathsf{S}}^*_{\mathsf{C}}(\mathsf{c}_{\sigma(i_m)})) \neq \mathsf{Dec}_{\mathsf{sk}}(\overline{\mathsf{S}}^*_{\mathsf{C}}(\mathsf{c}'_{\sigma'(i'_1)})) = \ldots \mathsf{Dec}_{\mathsf{sk}}(\overline{\mathsf{S}}^*_{\mathsf{C}}(\mathsf{c}'_{\sigma'(i'_m)})) \ .$$

However, by Claim 7.12, the above occurs with probability at most $\frac{2^{11}}{t}$. Combining this with the fact that $\bar{\mathsf{S}}^*$ wins the original split model game with probability at least ϵ, Corollary 7.13 follows. □

We can now deduce a lower bound on the expectation of $b + b'$, the winning indicators of $\mathcal{A}, A'$. Indeed:

$$\mathbb{E}[b + b'] \geq \Pr[\mathsf{G}]\mathbb{E}[\mathsf{b} + \mathsf{b}'|\mathsf{G}] \geq$$
$$\left(\epsilon - \frac{2^{11}}{t}\right)\left(\frac{2 + \frac{7}{4}t}{2 + 2t}\right) \geq \left(\epsilon - \frac{2^{11}}{t}\right)\frac{7}{16}$$

This concludes the proof of Proposition 7.11. □

Finally we conclude the proof of Proposition 7.7. Indeed, combining Proposition 7.8 and Proposition 7.11 we conclude that any adversary S^* cannot break Protocol 2 with probability greater than

$$\delta \leq \sqrt{2^{12}/t} - \mathrm{negl}(k)$$

. □

7.3 Amplifying Binding Using Parallel Repetition

The natural way to try and amplify the binding of our protocol is to repeat it in parallel, similarly to the soundness amplification done in [CKV10]. However, in our setting, this turns out to be much more challenging. Roughly, in [CKV10] there is an a-priori fixed function F, and the alleged homomorphic evaluations produced by the adversary can be compared independently to the preprocessed evaluation of F (on the dummy zero-encryptions). In our case, there is no explicit function F, instead, the committing adversary implicitly defines some function $\overline{\mathsf{S}}^*_{\mathsf{C}}$ that operates on ciphers in some arbitrary and possibly adversarial way, and the receiver compares the evaluation of $\overline{\mathsf{S}}^*_{\mathsf{C}}$ from the commitment phase to the alleged homomorphic evaluations produced in the decommitment phase. Unlike in [CKV10], in our case, when performing parallel repetition, the (possibly adversarial) function $\overline{\mathsf{S}}^*_{\mathsf{C}}$ now operates jointly on all of the dummy zero-encryptions; as a result, unexpected correlations may be produced.

To overcome the above problem, we will need to augment our protocol and then make use of parallel theorems for interactive arguments. The works of [Hai09] and [CL10] both show how to transform any weakly sound argument system to a new weakly sound system, for which parallel repetition is guaranteed to reduce the soundness error as required (in a weakly exponential rate). In a nutshell, both transformations ensure that there is an efficient way to sample random continuations of the verifier's messages in the protocol, for any given partial transcript, and without knowing the coins used to generate the verifier's messages in this partial transcript. This can already be shown to be sufficient for parallel repetition to work.

Since the above theorems work for interactive arguments, we wish to go through the following path:

1. Turn the weakly binding function commitment into a weakly sound protocol.
2. Apply one of the above transformation to obtain a strongly sound protocol.
3. Show that the strong soundness of the (parallel repeated) protocol implies the strong binding of the parallel repetition of the function commitment.

Difficulties in executing the above plan and their resolution. The standard way of applying parallel repetition theorems to standard weakly binding commitments schemes (rather than succinct function commitments) is to consider a proof system where the verifier V takes on the role of the receiver R and the prover P^* takes the role of the sender S^*. The parties perform the commit stage and then, in the decommitment stage, the prover should send the verifier two distinct values (y_0, y_1) together with their decommitment information; the verifier accepts only if the decommitments are valid (and indeed $y_0 \neq y_1$). Since the commitment is weakly binding, the protocol is weakly sound, and hence it can be transformed into a protocol that is amenable to soundness amplification via parallel repetition. For the transformations of [Hai09, CL10] it is also the case that the resulting protocol induces a strongly binding commitment scheme. Also, the described transformation has the benefit of preserving the round complexity of the commitment scheme.

Trying to apply the same reasoning to our SFC protocol does not work as is. The issue is that our decommitment phase involves a random challenge, and given two different decommitment challenges (with respect to the same commitment), the prover may cheat; indeed, recall that each random challenge includes the same zero encryption ez, and if the sender can identify it, he may equivocate. Hence, we need to consider a different protocol (or alternatively generalize the parallel repetition theorem, to the case of two non-communicating decommitters).

One option is to consider a protocol $(\mathsf{P}^*, \mathsf{V})$ where, after the commitment phase, the prover should send two strings (y_0, y_1) that he claims to be able to decommit to; then, the verifier picks a random $b \in \{0, 1\}$, and sends the prover the random decommitment challenge together with the bit b. The prover should then

provide a valid decommitment for y_b. It can be seen that if our SFC protocol $(\mathsf{S}^*, \mathsf{R})$ has binding error at most $1-\delta$, then the protocol $(\mathsf{P}^*, \mathsf{V})$ has soundness error at most $1-\frac{\delta}{2}$. Hence, we can transform the protocol $(\mathsf{P}^*, \mathsf{V})$ to a new protocol $(\mathsf{P}^*, \tilde{\mathsf{V}})$ whose parallel repetition $(\mathsf{P}^{(\ell)^*}, \tilde{\mathsf{V}}^{(\ell)})$ is strongly sound.

However, the protocol we have constructed is a six-message protocol (the decommitment phase now consists of four messages instead of two), and indeed, the corresponding strongly binding commitment will also be a six message protocol, where in the decommitment phase, first the query q is sent, then the result $f(q)$ is returned, and only then the sender is tested for its correctness (in particular, the value on which it chooses to be tested cannot depend on the challenge).

In order for our scheme to remain four-message, we first tweak the decommitment phase of our protocol as follows: instead of sending the challenge (the zero-encryption ez) in the clear, it is encrypted under a fresh key of the fully-homomorphic encryption scheme $\bar{\mathsf{sk}}$, and then the sender is required to homomorphically answer under the encryption. Indeed, now we can already consider the four round corresponding protocol, where the sender is given to decommitment challenges and is required to open them to different values. Unlike before, in this variant receiving two independent challenges (even for the same commitment phase) does not seem to admit any obvious attack, and indeed we manage to show that this protocol has weak soundness (related to the binding of the commitment).

In fact, we can show that this protocol already assures that random continuations of the verifier's messages can be easily simulated, and hence there is no need to additionally invoke the transformations of [Hai09] or [CL10]. As a matter of fact the tweak is already very similar to transformation of [CL10] that makes an arbitrary protocol "continuation simulatable". The difference is that [CL10] starts from an already (weakly) sound protocol (and maintains its weak soundness), while we will need to work in order to establish that the tweaked protocol is indeed sound. Details follow.

First, we show that Protocol 3 induces a weakly-sound four-message protocol. Then, we show that the protocol is "simulatable" and can hence be repeated in parallel to reduce the soundness error. Finally, we show that the the resulting strongly sound protocol induces a strongly binding commitment.

Proposition 7.14. *Let $(\tilde{\mathsf{S}}^*, \tilde{\mathsf{R}})$ be the function commitment given in 3. Consider the corresponding four-message protocol $(\mathsf{P}^*, \mathsf{V})$, where first P^* and V emulate the commitment phase of $\tilde{\mathsf{S}}^*$ and $\tilde{\mathsf{R}}$, then they execute the decommitment phase twice (with respect to the same commitment and the same query to the committed function), and V accepts only if the decommitments are valid and result in distinct values. Then, the protocol given by $(\mathsf{P}^*, \mathsf{V})$ is weakly sound.*

Proof sketch. Let $(\mathsf{S}^*, \mathsf{R})$ be the weakly binding commitment scheme given in 2, and recall that the tweaked protocol $(\tilde{\mathsf{S}}^*, \tilde{\mathsf{R}})$ is the same as $(\mathsf{S}^*, \mathsf{R})$, except that the decommitment phase is done under fully-homomorphic encryption (with a fresh key). We show that the protocol $(\mathsf{P}^*, \mathsf{V})$ described above (based on $(\tilde{\mathsf{S}}^*, \tilde{\mathsf{R}})$) has soundness error at most $3\delta^{\frac{1}{3}} + \mathrm{negl}(k)$, where δ is the binding-error of (the non-tweaked) $(\mathsf{S}^*, \mathsf{R})$. Recalling that $(\mathsf{S}^*, \mathsf{R})$ has binding error at most $\delta = O(t^{-\frac{1}{2}})$, we can then deduce that for a large enough constant t, $(\mathsf{P}^*, \mathsf{V})$ has soundness error at most $1 - \Omega(1)$ as required.

Assume towards contradiction there exists a (wlog deterministic) prover that, for infinitely many $k \in \mathbb{N}$ and $q \in \mathsf{Dom}_k$ breaks the soundness of the protocol on q with probability $3\delta'^{\frac{1}{3}}$, we construct a sender $\mathsf{S}^* = (\mathsf{S}^*_{\mathsf{C}}, \mathsf{S}^*_{\mathsf{D}}, \mathsf{S}^{*\prime}_{\mathsf{D}})$ that breaks the binding of $(\mathsf{S}^*, \mathsf{R})$ with probability $\delta' - \mathrm{negl}(k)$. $\mathsf{S}^*_{\mathsf{C}}$ emulates P^* in the first round, and randomly flips a bit $b \in \{0,1\}$ that decides on one of two strategies $(\mathsf{S}^*_{\mathsf{D}b}, \mathsf{S}^{*\prime}_{\mathsf{D}b})$ to play in the decommitment phase. Recall that the prover P^* performs two decommitments (in parallel); intuitively, the two strategies will deal with two possible behaviors of P^*: the first will benefit from a prover P^* that

Protocol 3

Let k be the security parameter and let $\mathcal{F}_k$ be a set of functions mapping $\{0,1\}^{s(k)}$ to $\{0,1\}^{s'(k)}$. Let $t \geq 2$ be some constant.

Main ingredient.

- A fully homomorphic encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$. (Can be a symmetric key scheme, in which case the cipher may include any required evaluation key.)

The commitment phase.

The receiver $\tilde{\mathsf{R}}_\mathsf{C}$:

- Samples two symmetric keys: $\mathsf{sk} \leftarrow \mathsf{Gen}(1^k)$ and $\tilde{\mathsf{sk}} \leftarrow \mathsf{Gen}(1^k)$.
- Samples a double encryption of 0^s under the keys $(\mathsf{sk}, \tilde{\mathsf{sk}})$: let $\mathsf{ez} \leftarrow \mathsf{Enc}_{\mathsf{sk}}(0^s)$ and let $a \leftarrow \mathsf{Enc}_{\tilde{\mathsf{sk}}}(\mathsf{ez})$.
- Sends a to S_C, and keeps the state $\mathsf{st}_\mathsf{R} = (\mathsf{sk}, \tilde{\mathsf{sk}}, \mathsf{ez})$.

The sender $\tilde{\mathsf{S}}_\mathsf{C}$:

- Given input function $f \in \mathcal{F}$, let Eval_f denote the deterministic function that, given a cipher c of an s bit message, computes $\mathsf{Eval}(f, \mathsf{c})$.
- Given the challenge a, computes $c := \mathsf{Eval}(\mathsf{Eval}_f, a)$.
  (That is, homomorphically evaluates the homomorphic evaluation of f.).
- Send c to R_C as the commitment.

The decommitment phase.

The receiver $\tilde{\mathsf{R}}_\mathsf{D}$:

- Given input query $q \in \{0,1\}^s$, samples t independent encryptions of q under the (inner) secret key sk, i.e. $(\mathsf{eq}_1, \ldots, \mathsf{eq}_t) \leftarrow \mathsf{Enc}_{\mathsf{sk}}(q)$.
- Samples a random permutation $\sigma\colon\ \{0,1,\ldots,t\} \to \{0,1,\ldots,t\}$.
- Computes $\bar{q} := \sigma(\mathsf{ez}, \mathsf{eq}_1, \ldots, \mathsf{eq}_t)$. (We denote by $\sigma(u_0, u_1, \ldots, u_t)$ the permuted values $(u_{\sigma(0)}, u_{\sigma(1)}, \ldots, u_{\sigma(t)})$.)
- Samples a fresh secret key $\bar{\mathsf{sk}} \leftarrow \mathsf{Gen}(1^k)$ and computes $\mathsf{e\bar{q}} \leftarrow \mathsf{Enc}_{\bar{\mathsf{sk}}}(\bar{q})$.
- Sends $\mathsf{e\bar{q}}$ to the sender S and keeps σ and $\bar{\mathsf{sk}}$.

The sender $\tilde{\mathsf{S}}_\mathsf{D}$:

- Given a cipher $\mathsf{e\bar{q}}$, computes $\hat{\mathsf{e\bar{q}}} = \mathsf{Eval}(g, \mathsf{e\bar{q}})$, where g is defined as follows: given a tuple $\bar{q} = (\mathsf{c}_0, \mathsf{c}_1, \ldots, \mathsf{c}_t)$, compute $d_i = \mathsf{Eval}(f, \mathsf{c}_i)$ for $i = 0, 1, \ldots, t$, and output $d = (d_0, d_1, \ldots, d_t)$.
- Sends $\hat{\mathsf{e\bar{q}}}$ to R_D.

The receiver $\tilde{\mathsf{R}}_\mathsf{D}$ - verification and output:

- Given the commitment c, the evaluated decommitment cipher $\hat{\mathsf{e\bar{q}}}$, and $\mathsf{sk}, \tilde{\mathsf{sk}}, \bar{\mathsf{sk}}, \sigma$.
- Decrypts the decommitment $(d_0, d_1, \ldots, d_t) = \mathsf{Dec}_{\bar{\mathsf{sk}}}(\hat{\mathsf{e\bar{q}}})$.
- Computes $\hat{\mathsf{ez}} = \mathsf{Dec}_{\tilde{\mathsf{sk}}}(c)$ and checks that $d_{\sigma(0)} = \hat{\mathsf{ez}}$.
- Checks that $\mathsf{Dec}_{\mathsf{sk}}(d_{\sigma(1)}) = \mathsf{Dec}_{\mathsf{sk}}(d_{\sigma(2)}) = \cdots = \mathsf{Dec}_{\mathsf{sk}}(d_{\sigma(t)})$.
- If one of these checks fails, output $v := \perp$; otherwise, output $v := \mathsf{Dec}_{\mathsf{sk}}(d_{\sigma(1)})$.

Figure 3: A weakly-binding SFC $(\tilde{\mathsf{S}}, \tilde{\mathsf{R}})$ that is amenable to parallel repetition.

significantly varies the value it decommits to in the first decommitment message; the second will benefit from P^* that is highly consistent in its first decommitment, i.e. it almost always opens to the same value.

We first describe the strategy for $b = 0$: here, both $\mathsf{S}^*_{\mathsf{D}0}$ and $\mathsf{S}^{*\prime}_{\mathsf{D}0}$ run the same (randomized) algorithm: given a decommitment challenge $\bar{q}$, it samples two FHE keys $(\bar{\mathsf{sk}}, \bar{\mathsf{sk}}')$ on its own, encrypts $\bar{q}$, under the first key $\bar{\mathsf{sk}}$, encrypts $\perp$ under the second key $\bar{\mathsf{sk}}'$, and emulates the decommitment phase of P^*; then, when it gets two evaluated decommitments, it decrypts them to obtain (d, d') and returns d as its decommitment.

We now describe the strategy for $b = 1$, $\mathsf{S}^*_{\mathsf{D}1}$, will run the same algorithm as $\mathsf{S}^*_{\mathsf{D}0}$ (and $\mathsf{S}^{*\prime}_{\mathsf{D}0}$), i.e. it uses the first decommitment to emulate its own decommitment (while simulating the second decommitment with an encryption of $\perp$). $\mathsf{S}^{*\prime}_{\mathsf{D}1}$ is similar to $\mathsf{S}^*_{\mathsf{D}1}$, only that it reverses the order of things, i.e. it uses the second decommitment to emulate its own decommitment (while simulating the first decommitment with an encryption of $\perp$).

We now analyze the success probability of S^*. For an execution of $(\mathsf{P}^*, \mathsf{V})$, let r_{C} be the coins used by the verifier to sample the messages for the commitment phase, let r_{D} be the coins used by the verifier to sample the decommitment challenges $(\bar{q}, \bar{q}')$ and the keys $(\bar{\mathsf{sk}}, \bar{\mathsf{sk}}')$ used to encrypt them. Then, with probability at least $\delta'^{\frac{1}{3}}$ over the choice of r_{C}, it holds that, with probability at least $2\delta'^{\frac{1}{3}}$ over the choice of r_{D}, the prover P^* convinces V to accept; we call such r_{C} "good".

We show that, for all (but finitely many) good r_{C}'s, the sender S^* that we have constructed breaks the binding of the commitment scheme (where R has uses r_{C} and the commitment phase and has input q) with probability $\delta'^{\frac{2}{3}} - \mathrm{negl}(k)$. Overall, S^* will break binding with probability at least $\delta' - \mathrm{negl}(k)$ as required.

Indeed, fix any good r_{C} and let T be the corresponding transcript of the commitment phase and let $\bar{q}$ denote the distribution of the decommitment challenge corresponding to r_{C}. We will denote by $\mathsf{P}^*_{\mathsf{T}}$ the prover initialized to the state induced by T. To measure how much the prover $\mathsf{P}^*_{\mathsf{T}}$ varies its behavior on the first the decommitment, we define $\alpha(r_{\mathsf{C}})$ to be the probability that the prover validly decommits to two different values given two independent encryptions of $(\bar{q}_1, \bar{q}_2)$ (each sampled according to $\bar{q}$):

$$\alpha(r_{\mathsf{C}}) := \Pr_{\substack{(\bar{q}_1, \bar{\mathsf{sk}}_1, \bar{\mathsf{sk}}_1') \\ (\bar{q}_2, \bar{\mathsf{sk}}_2, \bar{\mathsf{sk}}_2')}} \left[\begin{array}{r|c} (\mathsf{c}_1, \mathsf{c}_1') \leftarrow (\mathsf{Enc}_{\bar{\mathsf{sk}}_1}(\bar{q}_1), \mathsf{Enc}_{\bar{\mathsf{sk}}_1'}(\perp)) & \\ (\hat{\mathsf{c}}_1, \hat{\mathsf{c}}_1') \leftarrow \mathsf{P}^*_{\mathsf{T}}(\mathsf{c}_1, \mathsf{c}_1') & \\ (d_1, d_1') \leftarrow (\mathsf{Dec}_{\mathsf{sk}_1}(\hat{\mathsf{c}}_1), \mathsf{Dec}_{\mathsf{sk}_1'}(\hat{\mathsf{c}}_1')) & \\ (\mathsf{c}_2, \mathsf{c}_2') \leftarrow (\mathsf{Enc}_{\bar{\mathsf{sk}}_2}(\bar{q}_2), \mathsf{Enc}_{\bar{\mathsf{sk}}_2'}(\perp)) & \text{Both } (d_1, d_2) \text{ are valid decommitments} \\ (\hat{\mathsf{c}}_2, \hat{\mathsf{c}}_2') \leftarrow \mathsf{P}^*_{\mathsf{T}}(\mathsf{c}_2, \mathsf{c}_2') & \text{to two different values} \\ (d_2, d_2') \leftarrow (\mathsf{Dec}_{\mathsf{sk}_2}(\hat{\mathsf{c}}_2), \mathsf{Dec}_{\mathsf{sk}_2'}(\hat{\mathsf{c}}_2')) & \end{array} \right]$$

We first note that by the construction of $(\mathsf{S}^*_{\mathsf{D}0}, \mathsf{S}^{*\prime}_{\mathsf{D}0})$ and the definition of $\alpha(r_{\mathsf{C}})$, $(\mathsf{S}^*_{\mathsf{D}0}, \mathsf{S}^{*\prime}_{\mathsf{D}0})$ breaks the binding of the scheme with probability $\alpha(r_{\mathsf{C}})$. Next, we will show that $(\mathsf{S}^*_{\mathsf{D}1}, \mathsf{S}^{*\prime}_{\mathsf{D}1})$ breaks the scheme with probability $4\delta'^{2/3} - \alpha(r_{\mathsf{C}}) - \mathrm{negl}(k)$ (for large enough k). This will suffice to conclude that $(\mathsf{S}^*_{\mathsf{D}}, \mathsf{S}^{*\prime}_{\mathsf{D}})$, which applies one of the above strategies picked at random, will break binding with probability at least $2\delta'^{2/3} - \mathrm{negl}(k)$ as required.

Claim 7.15. *For any good r_{C}, $(\mathsf{S}^*_{\mathsf{D}1}, \mathsf{S}^{*\prime}_{\mathsf{D}1})$ breaks the scheme with probability $s(r_{\mathsf{C}}) \geq 4\delta'^{2/3} - \alpha(r_{\mathsf{C}}) - \mathrm{negl}(k)$ (over its own coin tosses).*

Proof. Recall that $\mathsf{S}^*_{\mathsf{D}1}$, given a challenge $\bar{q}_1$, decommits by sampling $(\bar{\mathsf{sk}}_1, \bar{\mathsf{sk}}_1')$ on its own, emulating the decommitment of P^* when given $(\mathsf{Enc}_{\bar{\mathsf{sk}}_1}(\bar{q}), \mathsf{Enc}_{\bar{\mathsf{sk}}_1'}(\perp))$, and returning the decryption d_1 of the first evaluation returned by P^*. $\mathsf{S}^{*\prime}_{\mathsf{D}1}$ does the same in a reverse order; more explicitly, given a challenge $\bar{q}_2$, it decommits by sampling $(\bar{\mathsf{sk}}_2, \bar{\mathsf{sk}}_2')$ on its own, emulating the decommitment of P^* when given $(\mathsf{Enc}_{\bar{\mathsf{sk}}_2}(\bar{q}_2)$, $\mathsf{Enc}_{\bar{\mathsf{sk}}_2'}(\perp))$, and returning the decryption d_2' of the second evaluation returned by P^*. Hence the probability that $(\mathsf{S}^*_{\mathsf{D}1}, \mathsf{S}^{*\prime}_{\mathsf{D}1})$ equivocates is given by:

$$s(r_{\mathsf{C}}) := \Pr_{\substack{(\bar{q}_1,\bar{\mathsf{sk}}_1,\bar{\mathsf{sk}}'_1)\\(\bar{q}'_2,\bar{\mathsf{sk}}_2,\bar{\mathsf{sk}}'_2)}}\left[\begin{array}{r} (\mathsf{c}_1,\mathsf{c}'_1)\leftarrow(\mathsf{Enc}_{\bar{\mathsf{sk}}_1}(\bar{q}_1),\mathsf{Enc}_{\bar{\mathsf{sk}}'_1}(\bot))\\ (\hat{\mathsf{c}}_1,\hat{\mathsf{c}}'_1)\leftarrow\mathsf{P}^*_{\mathsf{T}}(\mathsf{c}_1,\mathsf{c}'_1)\\ (d_1,d'_1)\leftarrow(\mathsf{Dec}_{\mathsf{sk}_1}(\hat{\mathsf{c}}_1),\mathsf{Dec}_{\mathsf{sk}'_1}(\hat{\mathsf{c}}'_1))\\ \hline (\mathsf{c}_2,\mathsf{c}'_2)\leftarrow(\mathsf{Enc}_{\bar{\mathsf{sk}}_2}(\bot),\mathsf{Enc}_{\bar{\mathsf{sk}}'_2}(\bar{q}'_2))\\ (\hat{\mathsf{c}}_2,\hat{\mathsf{c}}'_2)\leftarrow\mathsf{P}^*_{\mathsf{T}}(\mathsf{c}_2,\mathsf{c}'_2)\\ (d_2,d'_2)\leftarrow(\mathsf{Dec}_{\mathsf{sk}_2}(\hat{\mathsf{c}}_2),\mathsf{Dec}_{\mathsf{sk}'_2}(\hat{\mathsf{c}}'_2))\end{array}\;\middle|\;\begin{array}{c}\text{Both } (d_1,d'_2) \text{ are valid decommitments}\\ \text{to two different values}\end{array}\right]$$

First, we consider an hybrid experiment H where in the two parts of the experiment, P^* gets encryptions $(\mathsf{Enc}_{\bar{\mathsf{sk}}_1}(\bar{q}_1),\mathsf{Enc}_{\bar{\mathsf{sk}}'_1}(\bot))$ and $(\mathsf{Enc}_{\bar{\mathsf{sk}}_2}(\bar{q}_2),\mathsf{Enc}_{\bar{\mathsf{sk}}'_2}(\bar{q}'_2))$ correspondingly. Denote by $s_H(r_{\mathsf{C}})$ the probability that the decrypted decommitments d_1 and d'_2 in H are valid decommitments to different values:

$$s_H(r_{\mathsf{C}}) := \Pr_{\substack{(\bar{q}_1,\bar{\mathsf{sk}}_1,\bar{\mathsf{sk}}'_1)\\(\boxed{\bar{q}_2},\bar{q}'_2,\bar{\mathsf{sk}}_2,\bar{\mathsf{sk}}'_2)}}\left[\begin{array}{r} (\mathsf{c}_1,\mathsf{c}'_1)\leftarrow(\mathsf{Enc}_{\bar{\mathsf{sk}}_1}(\bar{q}_1),\mathsf{Enc}_{\bar{\mathsf{sk}}'_1}(\bot))\\ (\hat{\mathsf{c}}_1,\hat{\mathsf{c}}'_1)\leftarrow\mathsf{P}^*_{\mathsf{T}}(\mathsf{c}_1,\mathsf{c}'_1)\\ (d_1,d'_1)\leftarrow(\mathsf{Dec}_{\mathsf{sk}_1}(\hat{\mathsf{c}}_1),\mathsf{Dec}_{\mathsf{sk}'_1}(\hat{\mathsf{c}}'_1))\\ \hline (\mathsf{c}_2,\mathsf{c}'_2)\leftarrow(\mathsf{Enc}_{\bar{\mathsf{sk}}_2}(\boxed{\bar{q}_2}),\mathsf{Enc}_{\bar{\mathsf{sk}}'_2}(\bar{q}'_2))\\ (\hat{\mathsf{c}}_2,\hat{\mathsf{c}}'_2)\leftarrow\mathsf{P}^*_{\mathsf{T}}(\mathsf{c}_2,\mathsf{c}'_2)\\ (d_2,d'_2)\leftarrow(\mathsf{Dec}_{\mathsf{sk}_2}(\hat{\mathsf{c}}_2),\mathsf{Dec}_{\mathsf{sk}'_2}(\hat{\mathsf{c}}'_2))\end{array}\;\middle|\;\begin{array}{c}\text{Both } (d_1,d'_2) \text{ are valid decommitments}\\ \text{to two different values}\end{array}\right]$$

Then, by semantic security, it holds that $s(r_{\mathsf{C}}) \geq s_H(r_{\mathsf{C}}) - \mathrm{negl}(k)$ (for all but finitely many $k \in \mathbb{N}$). Indeed, this event can be tested only by knowing the keys $\bar{\mathsf{sk}}_1$ and $\bar{\mathsf{sk}}'_2$, which are chosen independently from $\bar{\mathsf{sk}}'_1$.

Next, we consider the probability $\beta_H(r_{\mathsf{C}})$ that in the hybrid experiment H, d_1 is a valid decommitment and (d_2, d'_2) are valid decommitments to two different values:

$$\beta_H(r_{\mathsf{C}}) := \Pr_{\substack{(\bar{q}_1,\bar{\mathsf{sk}}_1,\bar{\mathsf{sk}}'_1)\\(\bar{q}_2,\bar{q}'_2,\bar{\mathsf{sk}}_2,\bar{\mathsf{sk}}'_2)}}\left[\begin{array}{r} (\mathsf{c}_1,\mathsf{c}'_1)\leftarrow(\mathsf{Enc}_{\bar{\mathsf{sk}}_1}(\bar{q}_1),\mathsf{Enc}_{\bar{\mathsf{sk}}'_1}(\bot))\\ (\hat{\mathsf{c}}_1,\hat{\mathsf{c}}'_1)\leftarrow\mathsf{P}^*_{\mathsf{T}}(\mathsf{c}_1,\mathsf{c}'_1)\\ (d_1,d'_1)\leftarrow(\mathsf{Dec}_{\mathsf{sk}_1}(\hat{\mathsf{c}}_1),\mathsf{Dec}_{\mathsf{sk}'_1}(\hat{\mathsf{c}}'_1))\\ \hline (\mathsf{c}_2,\mathsf{c}'_2)\leftarrow(\mathsf{Enc}_{\bar{\mathsf{sk}}_2}(\bar{q}_2),\mathsf{Enc}_{\bar{\mathsf{sk}}'_2}(\bar{q}'_2))\\ (\hat{\mathsf{c}}_2,\hat{\mathsf{c}}'_2)\leftarrow\mathsf{P}^*_{\mathsf{T}}(\mathsf{c}_2,\mathsf{c}'_2)\\ (d_2,d'_2)\leftarrow(\mathsf{Dec}_{\mathsf{sk}_2}(\hat{\mathsf{c}}_2),\mathsf{Dec}_{\mathsf{sk}'_2}(\hat{\mathsf{c}}'_2))\end{array}\;\middle|\;\begin{array}{c}\text{All of } (d_1,d_2,d'_2) \text{ are valid decommitments}\\ \text{and } (d_2,d'_2) \text{ correspond to two different values}\end{array}\right]$$

We also consider the probability that the same event happens in another hybrid experiment H', where all the encryptions are sample according to $\bar{q}$ (and none are sampled according to $\bot$):

$$\beta_{H'}(r_{\mathsf{C}}) := \Pr_{\substack{(\bar{q}_1,\boxed{\bar{q}'_1},\bar{\mathsf{sk}}_1,\bar{\mathsf{sk}}'_1)\\(\bar{q}_2,\bar{q}'_2,\bar{\mathsf{sk}}_2,\bar{\mathsf{sk}}'_2)}}\left[\begin{array}{r} (\mathsf{c}_1,\mathsf{c}'_1)\leftarrow(\mathsf{Enc}_{\bar{\mathsf{sk}}_1}(\bar{q}_1),\mathsf{Enc}_{\bar{\mathsf{sk}}'_1}(\boxed{\bar{q}'_1}))\\ (\hat{\mathsf{c}}_1,\hat{\mathsf{c}}'_1)\leftarrow\mathsf{P}^*_{\mathsf{T}}(\mathsf{c}_1,\mathsf{c}'_1)\\ (d_1,d'_1)\leftarrow(\mathsf{Dec}_{\mathsf{sk}_1}(\hat{\mathsf{c}}_1),\mathsf{Dec}_{\mathsf{sk}'_1}(\hat{\mathsf{c}}'_1))\\ \hline (\mathsf{c}_2,\mathsf{c}'_2)\leftarrow(\mathsf{Enc}_{\bar{\mathsf{sk}}_2}(\bar{q}_2),\mathsf{Enc}_{\bar{\mathsf{sk}}'_2}(\bar{q}'_2))\\ (\hat{\mathsf{c}}_2,\hat{\mathsf{c}}'_2)\leftarrow\mathsf{P}^*_{\mathsf{T}}(\mathsf{c}_2,\mathsf{c}'_2)\\ (d_2,d'_2)\leftarrow(\mathsf{Dec}_{\mathsf{sk}_2}(\hat{\mathsf{c}}_2),\mathsf{Dec}_{\mathsf{sk}'_2}(\hat{\mathsf{c}}'_2))\end{array}\;\middle|\;\begin{array}{c}\text{All of } (d_1,d_2,d'_2) \text{ are valid decommitments}\\ \text{and } (d_2,d'_2) \text{ correspond to two different values}\end{array}\right]$$

We first note that by semantic security $\beta_H(r_{\mathsf{C}}) \geq \beta_{H'}(r_{\mathsf{C}}) - \mathrm{negl}(k)$. Moreover, the view of the prover, in each of the two independent parts of the experiment given by H' is exactly the same as its view in the protocol $(\mathsf{P}^*, \mathsf{V})$. In particular, recall that in the protocol, conditioned on good r_{C}, the P^* manages to open both commitments validly and to different values with probability at least $2{\delta'}^{\frac{1}{3}}$, hence it follows that: $\beta_{H'}(r_{\mathsf{C}}) \geq 4{\delta'}^{\frac{2}{3}}$.

Going back to the hybrid H, we now consider the probability that d_1 and d_2 are two valid decommitments to two different values:

$$\alpha_H(r_{\mathsf{C}}) := \Pr_{\substack{(\bar{q}_1,\bar{\mathsf{sk}}_1,\bar{\mathsf{sk}}'_1)\\(\bar{q}_2,\boxed{\bar{q}'_2},\bar{\mathsf{sk}}_2,\bar{\mathsf{sk}}'_2)}}\left[\begin{array}{r|l} (\mathsf{c}_1,\mathsf{c}'_1)\leftarrow(\mathsf{Enc}_{\bar{\mathsf{sk}}_1}(\bar{q}_1),\mathsf{Enc}_{\bar{\mathsf{sk}}'_1}(\perp)) & \\ (\hat{\mathsf{c}}_1,\hat{\mathsf{c}}'_1)\leftarrow\mathsf{P}^*_{\mathsf{T}}(\mathsf{c}_1,\mathsf{c}'_1) & \\ (d_1,d'_1)\leftarrow(\mathsf{Dec}_{\mathsf{sk}_1}(\hat{\mathsf{c}}_1),\mathsf{Dec}_{\mathsf{sk}'_1}(\hat{\mathsf{c}}'_1)) & \\ \hline (\mathsf{c}_2,\mathsf{c}'_2)\leftarrow(\mathsf{Enc}_{\bar{\mathsf{sk}}_2}(\bar{q}_2),\mathsf{Enc}_{\bar{\mathsf{sk}}'_2}(\boxed{\bar{q}'_2})) & \\ (\hat{\mathsf{c}}_2,\hat{\mathsf{c}}'_2)\leftarrow\mathsf{P}^*_{\mathsf{T}}(\mathsf{c}_2,\mathsf{c}'_2) & \\ (d_2,d'_2)\leftarrow(\mathsf{Dec}_{\mathsf{sk}_2}(\hat{\mathsf{c}}_2),\mathsf{Dec}_{\mathsf{sk}'_2}(\hat{\mathsf{c}}'_2)) & \end{array}\;\middle|\;\begin{array}{c}\text{Both } (d_1,d_2) \text{ are valid decommitments}\\ \text{to two different values}\end{array}\right],$$

where the boxes mark the difference from $\alpha_H(r_{\mathsf{C}})$. Again by semantic security we can deduce that $\alpha_H(r_{\mathsf{C}}) \leq \alpha(r_{\mathsf{C}}) + \mathrm{negl}(k)$. It is now left to note that:

$$s_H(r_{\mathsf{C}}) \geq \beta_H(r_{\mathsf{C}}) - \alpha_H(r_{\mathsf{C}}) \ ;$$

indeed, whenever d_1, d_2, d'_2 are valid decommitments and d_2, d'_2 correspond to two different values (as given by $\beta_H(r_{\mathsf{C}})$), and d_1 and d_2 are the same (as given by the complement of $\alpha_H(r_{\mathsf{C}})$, conditioned on the two being valid), then d_1, d'_2 are two valid decommitments to two different values (as required in $s_H(r_{\mathsf{C}})$).

Putting everything together:

$$\begin{aligned} s(r_{\mathsf{C}}) &\geq \\ s_H(r_{\mathsf{C}}) - \mathrm{negl}(k) &\geq \\ \beta_H(r_{\mathsf{C}}) - \alpha_H(r_{\mathsf{C}}) - \mathrm{negl}(k) &\geq \\ \beta_H(r_{\mathsf{C}}) - \alpha(r_{\mathsf{C}}) - \mathrm{negl}(k) &\geq \\ \beta_{H'}(r_{\mathsf{C}}) - \alpha(r_{\mathsf{C}}) - \mathrm{negl}(k) &\geq \\ 4{\delta'}^{\frac{2}{3}} - \alpha(r_{\mathsf{C}}) - \mathrm{negl}(k) & \end{aligned}$$

. □

This concludes the proof of Proposition 7.14. □

After showing that protocol $(\mathsf{P}^*, \mathsf{V})$ is weakly sound, it is left to show that it is amenable to parallel repetition and that its parallel repetition induces a strongly binding commitment:

Proposition 7.16. *Let* $(\mathsf{P}^{*(\ell)}, \mathsf{V}^{(\ell)})$ *be the* ℓ*-fold parallel repetition of* (P, V)*. Then, for* $\ell = \omega(\log k)$*,* $(\mathsf{P}^{*(\ell)}, \mathsf{V}^{(\ell)})$ *has negligible soundness error.*

Proposition 7.17. *Protocol* $(\mathsf{P}^{*(\ell)}, \mathsf{V}^{(\ell)})$ *induces a function commitment scheme whose binding error is the same as the soundness error of the protocol.*

To prove the first proposition we first recall the notion of "simulatable protocols"; intuitively, these are protocols where given some partial transcript of the protocol it is easy to simulate the rest of the prover's view, without being given the coins used by the verifier so far. There are a few variants of simulatability in the parallel repetition literature; here, we shall use the notion of computational simulatability introduced in [CL10] (which itself generalizes previous notions as the one in [HPWP10]).

Definition 7.18 (Simulatable Verifier [CL10, HPWP10])**.** *A verifier* V *for, an* m *round protocol, is said to be simulatalbe if, for every poly-size prover* P^**, there exists a PPT simulator* $\mathcal{S}$*, with the following guarantee. For every* j*-long partial interaction, consisting of verifier and prover coins and messages* $\{R_i^{\mathsf{V}}\}_{i=1}^{j}, \{M_i^{\mathsf{V}}\}_{i=1}^{j}, \{R_i^{\mathsf{P}^*}\}_{i=1}^{j}, \{M_i^{\mathsf{P}^*}\}_{i=1}^{j}$*, the simulator* $\mathcal{S}$*, given only* $\{M_i^{\mathsf{V}}\}_{i=1}^{j}, \{R_i^{\mathsf{P}^*}\}_{i=1}^{j}, \{M_i^{\mathsf{P}^*}\}_{i=1}^{j}$*, generates a simulated continuation* $\{\tilde{M}_i^{\mathsf{V}}\}_{i=j+1}^{m}, \{\tilde{R}_i^{\mathsf{P}^*}\}_{i=j+1}^{m}, \{\tilde{M}_i^{\mathsf{P}^*}\}_{i=j+1}^{m}$ *of* P^**'s view, which is computationally indistinguishable from a real continuation, i.e.:*

$$\begin{matrix} \{M_i^{\mathsf{V}}\}_{i=1}^{j} \cup \{\tilde{M}_i^{\mathsf{V}}\}_{i=j+1}^{m} \\ \{R_i^{\mathsf{P}^*}\}_{i=1}^{j} \cup \{\tilde{R}_i^{\mathsf{P}^*}\}_{i=j+1}^{m} \\ \{M_i^{\mathsf{P}^*}\}_{i=1}^{j} \cup \{\tilde{M}_i^{\mathsf{P}^*}\}_{i=j+1}^{m} \end{matrix} \approx_c \begin{matrix} \{M_i^{\mathsf{V}}\}_{i=1}^{m} \\ \{R_i^{\mathsf{P}^*}\}_{i=1}^{m} \\ \{M_i^{\mathsf{P}^*}\}_{i=1}^{m} \end{matrix} .$$

(The simulator is not required to simulate V*'s decision bit.)*

In [CL10] it is shown that, for any protocol with a simulatable verifier, parallel repetition reduces the soundness in an exponential rate.

Theorem 7.19 (parallel repetition of simulatable protocols [CL10] (simplified version))**.** *Let* (P, V) *be a protocol with soundness error* $1 - \Omega(1)$*, and assume* V *is a simulatable verifier. Then, for* $\ell = \omega(\log(k))$*, the* ℓ*-fold repetition* $(\mathsf{P}^{*(\ell)}, \mathsf{V}^{(\ell)})$ *has soundness error* $\mathrm{negl}(k)$*.*

The proof of Proposition 7.16 now follows rather directly.

Proof of Proposition 7.16. First we note that the protocol $(\mathsf{P}^*, \mathsf{V})$ has a simulatable verifier. Indeed, since the protocol has only four messages, it suffices to show that given any transcript corresponding to the first round (i.e. the commitment), it is possible to sample a computationally indistinguishable continuation of the verifier message in the second (decommitment) round. Indeed, since in the decommitment round the verifier simply sends two encryptions under fresh keys (independent of its coins in the first round), the messages can be simulated with an encryption of any arbitrary plaintext, e.g. $\perp$.

In addition, as shown in Proposition 7.14, the protocol $(\mathsf{P}^*, \mathsf{V})$ has soundness error $1 - \Omega(1)$; hence, by Theorem 7.19, protocol $(\mathsf{P}^{*(\ell)}, \mathsf{V}^{(\ell)})$ has negligible soundness error. □

It is left to show that $(\mathsf{P}^{*(\ell)}, \mathsf{V}^{(\ell)})$ induces a strongly binding commitment. The induced commitment is given in Figure 4.

Protocol 4

Let k be the security parameter and let $\mathcal{F}_k$ be a set of functions mapping $\{0,1\}^{s(k)}$ to $\{0,1\}^{s'(k)}$.

Main ingredient.

- The weakly binding function commitment $(\tilde{\mathsf{S}}, \tilde{\mathsf{R}})$ given in 3.

The scheme.

- $\tilde{\mathsf{S}}^{(\ell)}$ and $\tilde{\mathsf{R}}^{(\ell)}$ run ℓ parallel copies of $\tilde{\mathsf{S}}$ and $\tilde{\mathsf{R}}$.
- Let $(v_1, \ldots, v_\ell)$ be the decommitted values. If $v_1 = v_2 = \cdots = v_\ell \neq \perp$, output $\tilde{\mathsf{R}}^{(\ell)}$ outputs $v = v_1$; otherwise, it outputs $v = \perp$.

Figure 4: A strongly-binding SFC $(\tilde{\mathsf{S}}^{(\ell)}, \tilde{\mathsf{R}}^{(\ell)})$.

Proof of Proposition 7.17. We claim that protocol $(\tilde{\mathsf{S}}^{(\ell)}, \tilde{\mathsf{R}}^{(\ell)})$ given in Figure 4 is a strongly binding protocol. Indeed, given a malicious sender $(\mathsf{S}_{\mathsf{C}}^{*(\ell)}, \mathsf{S}_{\mathsf{D}}^{*(\ell)}, \mathsf{S}_{\mathsf{D}}^{*\prime(\ell)})$ that breaks the binding of the protocol with probability ϵ we can immediately deduce a prover $\mathsf{P}^{*(\ell)}$ that breaks the soundness of $(\mathsf{P}^{*(\ell)}, \mathsf{V}^{(\ell)})$ with the same probability ϵ. $\mathsf{P}^{*(\ell)}$ runs $\mathsf{S}_{\mathsf{C}}^{*(\ell)}$ in the commitment phase, and then in the decommitment phase, when it gets challenges $((\mathsf{e}\bar{\mathsf{q}}_1, \mathsf{e}\bar{\mathsf{q}}'_1), \ldots, (\mathsf{e}\bar{\mathsf{q}}_\ell, \mathsf{e}\bar{\mathsf{q}}'_\ell))$, it runs $\mathsf{S}_{\mathsf{D}}^{*(\ell)}$ on $(\mathsf{e}\bar{\mathsf{q}}_1, \ldots, \mathsf{e}\bar{\mathsf{q}}_\ell)$ and $\mathsf{S}_{\mathsf{D}}^{*\prime(\ell)}$ on $(\mathsf{e}\bar{\mathsf{q}}'_1, \ldots, \mathsf{e}\bar{\mathsf{q}}'_\ell)$, to obtain responses $(\hat{\mathsf{e}\bar{\mathsf{q}}}_1, \ldots, \hat{\mathsf{e}\bar{\mathsf{q}}}_\ell)$ and $(\hat{\mathsf{e}\bar{\mathsf{q}}}'_1, \ldots, \hat{\mathsf{e}\bar{\mathsf{q}}}'_\ell)$. Then it responds to V with $((\hat{\mathsf{e}\bar{\mathsf{q}}}_1, \hat{\mathsf{e}\bar{\mathsf{q}}}'_1), \ldots, (\hat{\mathsf{e}\bar{\mathsf{q}}}_\ell, \hat{\mathsf{e}\bar{\mathsf{q}}}'_\ell))$.

It is not hard to see that the induced view of S^* in the protocol is exactly as in a regular commitment and hence it will succeed equivocating on each one of the challenges with probability ϵ. □

Putting it all together: the final SMFC Scheme. We can now use the single message SFC scheme we have constructed to get a multi-function scheme SMFC by invoking the SFC scheme independently in parallel for each one of the functions, as shown in Proposition 7.4.

8 (Interactive) Succinct Arguments From MIPs And SMFCs

We show how to use a succinct multi-function commitment (SMFC) scheme (such as the one we constructed from FHE in Section 7) to transform any one-round succinct MIP of knowledge (such as the one we constructed in Section 5.3) into a succinct interactive argument of knowledge. The resulting arguments will be complexity-preserving as long as the underlying SMFC and MIP are (which is indeed the case for our constructions.)

As outlined in Section 3.3, the idea of the construction is to first make the prover commit to the functions representing each one of the MIP provers, and only after that the verifier will send to the prover the queries for these provers (in the clear).

The construction is described in Figure 5.

Theorem 8.1. *The compiler given by Protocol 5 gives a private-coin succinct argument of knowledge for the universal relation $\mathcal{R}_{\mathcal{U}}$. Moreover:*

- *The round complexity of the succinct argument is the same as that of the function commitment.*
- *If the function commitment and MIP are complexity-preserving so is the resulting succinct argument.*

We now turn to the proof of Theorem 8.1. The completeness of the protocol follows directly from the completeness of the SMFC scheme and the completeness of the MIP protocol. The proof will focus on showing that the protocol is indeed an argument of knowledge; then, in Remark 8.2, we will discuss its succinctness and complexity preservation.

Proof sketch of Theorem 8.1. We first show soundness, and then explain how to also show proof of knowledge.

Assume towards contradiction that $\mathcal{P}^*$ is a malicious prover that convinces the verifier $\mathcal{V}$ of accepting infinitely may false statements $y = (M, x, t)$, for infinitely many $k \in \mathbb{N}$, with noticeable probability $\varepsilon = \varepsilon(k)$. We shall derive a contradiction to the soundness of the underlying MIP protocol.

Let $\mathsf{S}^* = \mathsf{S}^*_{\mathcal{P}^*}$ be the commitment sender induced by $\mathcal{P}^*$. For commitment coins (r_1, s_1) for S^* and R, let $\mathsf{Val}^{\mathsf{S}^*}_{r_1,s_1}$ be the circuit that, given queries $(q_1, \ldots, q_\ell)$, decommitment coins (r_2, s_2), and $i \in [\ell]$, emulates the corresponding commitment and decommitment phases and outputs the i-th decommitment value output by the receiver R (which may be $\perp$).

Protocol 5

Ingredients.

- A succinct MIP of knowledge (G, P, V) for $\mathcal{R}_{\mathcal{U}}$, where honest provers are from a family $\mathcal{F}$.
- A succinct multi-function commitment scheme (R, S) for $\mathcal{F}$.

Inputs.

- Prover $\mathcal{P}$'s input: 1^k and $(y, w) \in \mathcal{R}_{\mathcal{U}}$, where k is the security parameter, $y = (M, x, t)$.
- Verifier $\mathcal{V}$'s input: 1^k and y.

Commitment.

- The prover $\mathcal{P}$ runs $P(y, w, 1^k)$ to deduce the MIP prover functions $(P_1, \ldots, P_\ell)$ for (y, w). [a]
- The prover $\mathcal{P}$ and verifier $\mathcal{V}$ execute the commitment phase of the SMFC protocol (R, S), during which $\mathcal{P}$ commits to the functions $(P_1, \ldots, P_\ell)$.

Decommitment to queries.

- The verifier $\mathcal{V}$ runs the query generator $G(t, 1^k)$ to obtain ℓ queries $\vec{q} = (q_1, \ldots, q_\ell)$.
- The prover $\mathcal{P}$ and verifier $\mathcal{V}$ execute the decommitment phase of (S, R), where $\mathcal{V}$ requests decommitments to the queries $\vec{q}$. In case the decommitment phase fails, the verifier rejects. Otherwise, it obtains ℓ answers $\vec{a} = (a_1, \ldots, a_\ell)$.
- Assuming the decommitment phase was valid, $\mathcal{V}$ runs the MIP verifier $V(\vec{q}, y, \vec{a}, 1^k)$ and accepts only if V does.

[a] For complexity-preservation, $(P_1, \ldots, P_\ell)$ should be represented as required by the function commitment (in our case, a circuit computable in time t and space s).

Figure 5: Succinct interactive argument for the relation $\mathcal{R}_{\mathcal{U}}$ from a one-round succinct MIP of knowledge and a SMFC protocol.

Recall that by Claim 7.3 the binding property of the SMFC scheme implies that, for all large enough $k \in \mathbb{N}$, and any distribution $\mathcal{Q}$ on tuples of ℓ queries, there exist functions $(\tilde{f}_1, \ldots, \tilde{f}_\ell)$ such that:

$$\Pr_{r_1,s_1}\left[\Pr_{r_2,s_2,\vec{q}}\left[\exists i \in [\ell] : \mathsf{Val}^{\mathsf{S}^*}_{r_1,s_1}(\vec{q}; r_2, s_2; i) \notin \left\{\tilde{f}_i(r_1, s_1; q_i), \perp\right\}\right] \geq \frac{\varepsilon}{4}\right] \leq \mathrm{negl}(k) \ , \tag{8}$$

where r_1, r_2 are random coins for R, s_1, s_2 are random coins for S^*, and $\vec{q}$ is sampled from $\mathcal{Q}$.

By our assumption on $\mathcal{P}^*$ and a standard averaging argument, it holds that, with probability at least $\varepsilon/2$ over the coins r_1, s_1 used in the commitment phase, $\mathcal{P}^*$ convinces $\mathcal{V}$ of accepting the false statement y with probability at least $\varepsilon/2$ over the coins used in the second phase. Combining this with Equation (8), and letting $\mathcal{Q} = G(t, 1^k)$ be the query distribution generated by the MIP generator, we deduce that there exist coins (r_1, s_1, r_2, s_2), such that with probability at least $\varepsilon/4$ over the choice of $\vec{q} \leftarrow G(t, 1^k)$, S^* provides answers $(a_1, \ldots, a_\ell)$ according to $(\tilde{f}_1(r_1, s_1, \cdot), \ldots, \tilde{f}_\ell(r_1, s_1, \cdot))$ and $V(\vec{q}, y, \vec{a}, 1^k)$ accepts, implying that the polysize family of circuits $\{\mathsf{Val}_{r_1,s_1}(\cdot; r_2, s_2; i)\}_{i\in[\ell]}$ violates the soundness of the underlying MIP protocol.

The above proof, in fact, yields an extraction strategy which establishes the knowledge property. Indeed, an extractor $\mathcal{E}$ for $\mathcal{P}^*$ simply samples random coins r_1, s_1, r_2, s_2 and then runs the MIP extractor E with the polysize family of circuits $\mathsf{Val} = \{\mathsf{Val}_{r_1,s_1}(\cdot; r_2, s_2; i)\}_{i\in[\ell]}$. For all large enough k, it holds that with probability $\Omega(\varepsilon)$ over (r_1, s_1, r_2, s_2) it holds that with probability at least $\Omega(\varepsilon)$ over $\vec{q} \leftarrow G(t, 1^k)$, V is convinced by Val, and hence the MIP extractor extracts a witness with probability $\Omega(\varepsilon \cdot p_E(\varepsilon))$, where $p_E(\varepsilon)$ is the extraction probability guaranteed by E. □

Remark 8.2 (succinctness and complexity-preservation)**.** The succinctness of the verifier in the above construction follows readily from the succinctness of the underlying MIP verifier, as well as the succinctness of the receiver in the underlying SMFC scheme. Moreover, assuming that both the MIP and SMFC are complexity preserving, so is the resulting succinct argument. More accurately, we should make sure that the representation of the MIP prover functions is such that the SMFC is indeed complexity preserving, which is indeed the case in our construction, where the provers can be represented by appropriate succinct circuits.

Remark 8.3 (universality and weak extraction)**.** The extractor $\mathcal{E}$ described above, on input $y = (M, x, t)$ and with oracle access to $\mathcal{P}^*$, may run as long as $\mathrm{poly}(t)$. However, because the MIP protocol we construct in Section 6 also provides *local* extraction guarantees (as it uses proximity testing to locally-decodable Reed–Muller codes), the extraction guarantee can be generalized to the *weak proof of knowledge property* as defined in [BG08] (see Remark 5.9). According to this notion, the extractor produces an implicit circuit representation of the witness, which allows to extract each particular bit in polynomial time, regardless of t.

Remark 8.4 (adaptivity)**.** As stated, our protocol is non-adaptive, in the sense that the instance y is given to the verifier at the beginning of the protocol, rather than being adaptively chosen by the prover during the protocol. Of course, if one is willing to add a message where the instance y is sent by the prover, the this is a non-issue.

Nonetheless, we can obtain a restricted form of adaptivity, where the instance is adaptively chosen by the prover based on the commitment stage, and then sent to the prover in the beginning of the decommitment phase; this form of adaptivity is similar to the one that can be obtained in the construction of Kilian [Kil92].

9 Homomorphism-Extractable Encryption Schemes

Let $\mathcal{F}$ be a set of functions. An *homomorphism-extractable encryption* (HEE) scheme $\mathsf{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ for $\mathcal{F}$ is an $\mathcal{F}$-homomorphic (semantically-secure) encryption scheme where, very informally, homomorphic operations cannot be performed "obliviously".

The tuple of algorithms works as follows:

- Gen is a key generation algorithm that, given the security parameter 1^k, outputs secret and public keys $(\mathsf{sk}, \mathsf{pk})$; we shall denote by $\mathsf{Gen}(1^k, \ell)$, ℓ independent applications of $\mathsf{Gen}(1^k)$;
- Enc is an encryption algorithm and Dec is a decryption algorithm; and
- Eval is a homomorphic evaluation algorithm that gets a function $f \in \mathcal{F}$ and a ciphertext c and homomorphically evaluates the function on the underlying plaintext.

When decrypting an evaluated ciphertext $\hat{\mathsf{c}}$, the decryption algorithm also gets a "reference cipher" c, which we denote as $\mathsf{Dec}^{\mathsf{c}}_{\mathsf{sk}}(\hat{\mathsf{c}})$, and outputs either a decryption of $\hat{\mathsf{c}}$ or $\perp$ if $\hat{\mathsf{c}}$ is "invalid" with respect to c. Of course, we require that if $\hat{\mathsf{c}}$ is the result of an honest evaluation of a function $f \in \mathcal{F}$ on $\mathsf{c} = \mathsf{Enc}_{\mathsf{pk}}(m)$, then $\mathsf{Dec}^{\mathsf{c}}_{\mathsf{sk}}(\hat{\mathsf{c}}) = m$.

The homomorphism-extractable property essentially says that an adversarial evaluator (or several colluding evaluators) that are given a tuple of encryptions $(\mathsf{Enc}_{\mathsf{pk}_1}(m_1), \ldots, \mathsf{Enc}_{\mathsf{pk}_\ell}(m_\ell))$, cannot produce a valid tuple of (allegedly) evaluated ciphers $(\hat{\mathsf{c}}_1, \ldots, \hat{\mathsf{c}}_\ell)$, without "knowing" a tuple functions $(\mathsf{C}_1, \ldots, \mathsf{C}_\ell)$ such that each $\hat{\mathsf{c}}_i$ is the result of homomorphically evaluating C_i on $\mathsf{Enc}_{\mathsf{pk}_i}(m_i)$.

Definition 9.1. *Let* $\mathsf{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ *be a public-key encryption scheme that is homomorphic with respect to a set of functions* $\mathcal{F}$*. We say that* E *is a* **homomorphism-extractable encryption** *(HEE) scheme if the following conditions hold:*

1. **Correctness.** *For every message* m *and function* $f \in \mathcal{F}$*:*

$$\Pr\left[\begin{array}{c|c} (\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{Gen}(1^k) & \hat{\mathsf{c}} \leftarrow \mathsf{Eval}_{\mathsf{pk}}(f,\mathsf{c}) \\ \mathsf{c} \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m) & \mathsf{Dec}^{\mathsf{c}}_{\mathsf{sk}}(\hat{\mathsf{c}}) = m \end{array}\right] \geq 1 - \mathrm{negl}(k) \ .$$

2. **Semantic security.** *For every poly-size adversary* $\mathcal{A}$*, large enough* $k \in \mathbb{N}$*, and two messages* $m_0, m_1 \in \{0,1\}^{\mathrm{poly}(k)}$*:*

$$\Pr\left[\mathcal{A}(c_0) = 1 \;\middle|\; \begin{array}{c} (\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{Gen}(1^k) \\ \mathsf{c}_0 \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m_0) \end{array}\right] - \Pr\left[\mathcal{A}(c_1) = 1 \;\middle|\; \begin{array}{c} (\mathsf{sk},\mathsf{pk}) \leftarrow \mathsf{Gen}(1^k) \\ \mathsf{c}_1 \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m_1) \end{array}\right] \leq \mathrm{negl}(k) \ .$$

3. **Homomorphism extractability.** *For any poly-size adversary* $\mathcal{A}$ *there exists a poly-size extractor* $\mathcal{E}^{\mathsf{E}}_{\mathcal{A}}$ *such that for all large enough security parameter* $k \in \mathbb{N}$*, auxiliary input* $z \in \{0,1\}^{\mathrm{poly}(k)}$*, and any efficiently samplable distribution* $\mathcal{M}_\ell$ *on tuples of* $\ell = \mathrm{poly}(k)$ *plaintexts:*

$$\Pr\left[\begin{array}{c|c} (\vec{\mathsf{sk}},\vec{\mathsf{pk}}) \leftarrow \mathsf{Gen}(1^k,\ell) & (\hat{\mathsf{c}}_1,\ldots,\hat{\mathsf{c}}_\ell) \leftarrow \mathcal{A}(\mathsf{c}_1,\ldots,\mathsf{c}_\ell;\vec{\mathsf{pk}},z) \\ (m_1,\ldots,m_\ell) \leftarrow \mathcal{M}_\ell & (\mathsf{C}_1,\ldots,\mathsf{C}_\ell) \leftarrow \mathcal{E}^{\mathsf{E}}_{\mathcal{A}}(\mathsf{c}_1,\ldots,\mathsf{c}_\ell;\vec{\mathsf{pk}},z) \\ \forall\, i \in [\ell] : \mathsf{c}_i \leftarrow \mathsf{Enc}_{\mathsf{pk}_i}(m_i) & \exists\, i \in [\ell] : \mathsf{Dec}^{\mathsf{c}_i}_{\mathsf{sk}_i}(\hat{\mathsf{c}}_i) \notin \{\mathsf{C}_i(m_i), \perp\} \end{array}\right] \leq \mathrm{negl}(k) \ .$$

4. **Succinctness.** *For any function* $f \in \mathcal{F}$*, with running time* $t(n)$ *and output size* $r(n)$ *on inputs of size* n*, and and any ciphertext* c *encrypting a plaintext of size* n*, computing* $\mathsf{Eval}(f,\mathsf{c})$ *requires* $\mathrm{poly}(k, t(n))$ *time and its result* $\hat{\mathsf{c}}$ *has size* $\mathrm{poly}(r(n)+k)$*. (Also see Remark 5.1.)*

   Moreover, the decryption $\mathsf{Dec}^{\mathsf{c}}_{\mathsf{sk}}(\hat{\mathsf{c}})$ *of an (allegedly) evaluated ciphertext* $\hat{\mathsf{c}}$*, with respect to an original ciphertext* c*, takes time at most* $\mathrm{poly}(|\mathsf{c}| + |\hat{\mathsf{c}}| + k)$ *(which is, in particular, independent of* $t(n)$*).*

9.1 Remarks And Discussion

Succinctness. A non-succinct HEE scheme can be trivially constructed by simply letting the evaluator algorithm append to the input ciphertext the description of the function to be evaluated. However, in our main application (namely, constructing SNARKs) the succinctness property is crucial: it ensures that the evaluator algorithm works not much harder than evaluating the function in the clear, that the evaluated ciphertext is not much longer than the output size of the function, and that the decryption algorithm runs in time that is only polynomial in the evaluated ciphertext (and the "reference" ciphertext); in particular, the decryption algorithm runs in time that is independent of the time required to evaluate the function.

On auxiliary input. Definition 9.1 requires that, for any adversary auxiliary input $z \in \{0,1\}^{\mathrm{poly}(k)}$, the poly-size extractor succeeds when given the same auxiliary input z. This requirement seems rather strong considering the fact that z could potentially encode arbitrary circuits. For example, the auxiliary input z may encode a circuit that, given the public key pk as input, outputs $\mathsf{Enc}_{\mathsf{pk}}(x)$ where $x = f_s(\mathsf{pk})$ is the image of some hardwired pseudorandom function f_s. In this case, the extractor would essentially be required to (efficiently) reverse engineer the circuit, which seems to be a rather strong requirement (or even an impossible one, under certain obfuscation assumptions).

While for presentational purposes the above definition may be simple and convenient, it can be relaxed to specific "benign" auxiliary inputs distributions; indeed, in our application, it will be sufficient to only consider a truly random auxiliary input z. (Requiring less than that, seems to be not expressive enough, as we would at least like to allow the adversary to toss random coins, and accordingly provide the extractor with the same random coins).

Plaintext distributions. The homomorphism-extractability property is formulated for all plaintext distributions $\mathcal{M}_\ell$; however, it can be relaxed to deal with specific distributions on plaintexts, e.g., ones that have high min-entropy. (In our application, each plaintext in a tuple sampled from $\mathcal{M}_\ell$ will be uniformly random, but different coordinates may be correlated.)

"Stand-alone" HEE. The homomorphism-extractability property can be restricted to the case of *single-plaintext distributions* (i.e. $\ell = 1$). The multi-ciphertext notion can be shown to follow from the single-ciphertext "stand-alone" one, provided that we assume that the latter holds with respect to appropriate auxiliary input distributions. (For instance, if the property holds for any single-plaintext distribution $\mathcal{M}_1$ and for any auxiliary input $z \in \{0,1\}^{\mathrm{poly}(k)}$, then it also holds for any distribution of multiple plaintexts $\mathcal{M}_\ell$, as in Definition 9.1.)

Sparsity. A feature shared by HEE encryption and other extractable primitives is the need for "structure" that prevents "oblivious attacks". In the case of HEE, it should be hard for any adversary to obliviously sample valid evaluations of a ciphertext without any knowledge of a function relating the two plaintexts.

Specifically, for a given ciphertext c and corresponding public key pk, consider the function $I_{\mathsf{pk},\mathsf{c}} := \mathsf{Eval}_{\mathsf{pk}}(\cdot,\mathsf{c})\colon \mathcal{F} \to \{0,1\}^r$ that maps each $f \in \mathcal{F}$ to an (honestly) evaluated ciphertext $\hat{\mathsf{c}}$. To a first approximation (and temporarily being inaccurate), the homomorphism-extractability property requires that whenever the adversary $\mathcal{A}$, given $(\mathsf{pk}, \mathsf{c}_1, \ldots, \mathsf{c}_\ell)$ where $\mathsf{c}_i = \mathsf{Enc}_{\mathsf{pk}}(m_i)$, manages to output $y \in \mathsf{Image}(I_{\mathsf{pk},\mathsf{c}_i})$ for some i, the extractor must provide a "pre-image" $f_i \in \mathcal{F}$. In general, for some distributions on $\mathsf{Image}(I_{\mathsf{pk},\mathsf{c}_i})$ this may be a hard task; thus, any HEE scheme candidate should not enable the adversary from being able to obliviously sample from such a distribution (i.e., without actually applying $I_{\mathsf{pk},\mathsf{c}_i}$ to some input).

To a second approximation (and now being accurate), in the previous paragraph, the condition for the extractor to work is too weak and the requirement on the output of the extractor is too strong: the extractor must work not only when the adversary manages to output $y \in \mathsf{Image}(I_{\mathsf{pk},\mathsf{c}_i})$ but whenever the adversary

outputs a valid ciphertext (namely, one where $\mathsf{Dec}^{\mathsf{c}_i}_{\mathsf{sk}_i}(y) \neq \perp$); moreover, the extractor is not required to output an "explanation" $f_i \in \mathcal{F}$ but may instead output any function C_i such that $\mathsf{C}_i(m_i) = \mathsf{Dec}^{\mathsf{c}_i}_{\mathsf{sk}_i}(y)$ (so that, in particular, an HEE scheme has no integrity guarantees).

As in the case of extractable one-way or hash functions (see discussion in [BCCT12a]), this issue is dealt with by considering candidate constructions where the image of $I_{\mathsf{pk},\mathsf{c}_i}$ is super-polynomially sparse within its range $\{0,1\}^r$, with overwhelming probability over $(\mathsf{pk}, \mathsf{c})$.

Of course, ensuring such a sparsity property merely satisfies one necessary condition, and potentially may be a long way off from actually ensuring homomorphism extractability. Still, ensuring sparsity rules out one of the few generic attacks about which we can reason without venturing into the poorly-charted territory of non-black-box extraction.

Targeted non-malleability. The extractable-homomorphism property does *not* provide any guarantee regarding the extracted functions C_i (e.g, they may not be in the family $\mathcal{F}$). In particular, the property does not ensure targeted malleability in the sense of, e.g., [BSW12]. (This also suggests the plausibility of rather simple and direct candidate constructions, as the one described in Section 9.2.) One could strengthen the definition to add this requirement as well (which is not required in our application), by replacing the requirement on the extracted circuits C_i from

$$\mathsf{Dec}^{\mathsf{c}_i}_{\mathsf{sk}_i}(\hat{\mathsf{c}}_i) \notin \{\mathsf{C}_i(m_i), \perp\}$$

to

$$\mathsf{Dec}^{\mathsf{c}_i}_{\mathsf{sk}_i}(\hat{\mathsf{c}}_i) \notin \left\{\mathsf{Dec}^{\mathsf{c}_i}_{\mathsf{sk}_i}(\mathsf{Eval}(\mathsf{C}_i, \mathsf{c}_i)), \perp\right\} \quad .$$

9.2 Candidate Constructions

We briefly discuss the plausibility of the existence of HEE schemes.

From ECRHs and FHE. Leaving efficiency aside and purely from an existential perspective, HEE schemes exist as long as extractable collision-resistant hashes (ECRHs) and FHE schemes exist: [BCCT12a] showed how to construct SNARKs from ECRHs, and in Section 10.3 we will show that SNARKs and FHE together imply HEE schemes.

However, because our motivation for studying HEE schemes is the construction of SNARKs that are potentially more efficient than those built using PCPs (as in [BCCT12a]), we believe that a more interesting question is to understand how plausible are "direct" constructions of HEE schemes.

Direct constructions. Recall that an extractability assumption asserts that exhibiting a certain kind of behavior cannot be done without "knowing" certain information; in particular, such an assumption rules out adversaries attempting to carry out "oblivious attacks". In the specific case of HEE schemes, the homomorphism-extractable property asserts that it is not possible for an efficient adversary to homomorphically evaluate ciphertexts without being aware of what operations are being applied to the underlying plain texts.

Of course, if an encryption scheme has *no* form of homomorphism, then it is certainly an HEE scheme, for $\mathcal{F} = \emptyset$. The tension arises in that we wish the encryption scheme to simultaneously support homomorphism but rule out "oblivious homomorphism".

Can we find, say, "direct" constructions of *fully*-homomorphic encryption schemes that can be plausibly assumed to be FHE schemes?

Consider, for example, the scheme of [BV11], based on LWE; in their scheme, an adversary, given as input a ciphertext c, may choose to ignore running the evaluation algorithm and instead simply apply some

sequence of algebraic operations (e.g., squaring the ciphertext) that likely will produce a new ciphertext $\hat{\mathsf{c}}$ that is *still* valid, and yet the adversary may not be aware of any function that maps the decryption of c to the decryption of $\hat{\mathsf{c}}$.

A method that appears to rule out the adversary in the previous paragraph is to consider an "amplified" version of the encryption scheme, where the ciphertext space has become sparse in its range so that oblivious attacks such as the above are going to result in invalid ciphertexts.

The idea of amplification is simple:

Construction 9.2 ("amplification")**.** Let $\ell \in \mathbb{N}$ and $\mathsf{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ be an $\mathcal{F}$-homomorphic encryption scheme. Construct a new $\mathcal{F}$-homomorphic encryption scheme $\tilde{\mathsf{E}}$, which we call the ℓ-amplification of E, as follows:

- The new generation algorithm $\tilde{\mathsf{Gen}}$, given input 1^k, runs $\mathsf{Gen}(1^k)$ with independent randomness for ℓ times to obtain ℓ pairs of secret and a public encryption keys, and then sets $(\tilde{\mathsf{sk}}, \tilde{\mathsf{pk}}) = ((\mathsf{sk}_1, \ldots, \mathsf{sk}_\ell), (\mathsf{pk}_1, \ldots, \mathsf{pk}_\ell))$.
- The new encryption algorithm $\tilde{\mathsf{Enc}}_{\tilde{\mathsf{pk}}}$, on input a plaintext m, independently encrypts m under each public key pk_i to obtain ℓ ciphertexts $\mathsf{c}_1, \ldots, \mathsf{c}_\ell$, and then outputs $\tilde{\mathsf{c}} := (\mathsf{c}_1, \ldots, \mathsf{c}_\ell)$.
- The new homomorphic evaluation algorithm $\tilde{\mathsf{Eval}}_{\mathsf{pk}}$, on input ℓ ciphertexts $\tilde{\mathsf{c}} = (\mathsf{c}_1, \ldots, \mathsf{c}_\ell)$ and a function $f \in \mathcal{F}$, computes $\hat{\mathsf{c}}_i := \mathsf{Eval}_{\mathsf{pk}_i}(f, \mathsf{c}_i)$ and then outputs $\hat{\tilde{\mathsf{c}}} := (\hat{\mathsf{c}}_1, \ldots, \hat{\mathsf{c}}_\ell)$.
- The new decryption algorithm $\tilde{\mathsf{Dec}}_{\tilde{\mathsf{sk}}}(\hat{\mathsf{c}})$, on input ℓ ciphertexts $\tilde{\mathsf{c}} = (\mathsf{c}_1, \ldots, \mathsf{c}_\ell)$, outputs $(\mathsf{Dec}_{\mathsf{sk}_1}(\mathsf{c}_1), \ldots, \mathsf{Dec}_{\mathsf{sk}_\ell}(\mathsf{c}_\ell))$.

While we do not have theoretical evidence to offer in this direction, we conjecture that amplification is a good heuristic for transforming an FHE scheme into another one that can plausibly be assumed to have the extractable-homomorphism property. It is an interesting question to formalize this intuition, and we leave this to future work.

10 From Homomorphism-Extractability To SNARKs And Back Again

In this section, we show how to SNARKs from succinct MIPs and homomorphism-extractable encryption (HEE). The resulting SNARKs will be complexity-preserving as long as the underlying MIP is, and the HEE evaluation algorithm is local (which is indeed the case for our constructions.) We also show that HEE schemes are, in fact, implied by SNARKs and standard homomorphic encryption.

10.1 Construction Details

The detailed construction of the protocol is presented in Figure 6. It will be useful to review Definition 5.13.

We shall prove the following theorem:

Theorem 10.1. *For any $c \in \mathbb{N}$, the construction in Figure 6 gives a SNARK for $\mathcal{R}_c$. Moreover, assuming that the MIP is complexity-preserving, the the SNARK is also complexity-preserving.* [19]

[19] Both Remarks 3.4 and 7.6 apply for this theorem; that is, to obtain complexity preservation, we require that MIP prover has a sufficiently tight *deterministic* reduction to the an appropriate circuit representation, which is indeed the case for our MIP construction. We also require that the homomorphism-extractable FHE scheme has a "local" evaluation algorithm, which is the case in all known FHE schemes.

Ingredients.

- A one-round succinct MIP of knowledge (G, P, V, E) for $\mathcal{R}_c$, where honest provers are functions from a family $\mathcal{F}$.
- A (symmetric-key) HEE scheme $\mathsf{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ for $\mathcal{F}$.

Setup $\mathcal{G}(1^k)$.

- Generate (private) verification state:
  - Set the time bound $B := k^{\log k}$. Sample $\ell := \ell(k)$ MIP queries $(q_1, \ldots, q_\ell) \leftarrow G(B, 1^k)$.
  - Sample ℓ secret keys for the HEE scheme $(\mathsf{sk}_1, \ldots, \mathsf{sk}_\ell) \leftarrow \mathsf{Gen}(1^k, \ell)$.
  - Set the verification state to be $\tau := (\vec{q}, \vec{\mathsf{sk}})$.
- Generate corresponding (verifier-generated) reference string:
  - Compute encryptions of each of the MIP queries $(\mathsf{c}_1, \ldots, \mathsf{c}_\ell) \leftarrow (\mathsf{Enc}(q_1), \ldots, \mathsf{Enc}(q_\ell))$.
  - Set the reference string to be $\sigma := \vec{\mathsf{c}}$.

Proof generation by $\mathcal{P}$.

- Input: $1^k, \sigma, (y, w) \in \mathcal{R}_c$ where $y = (M, x, t)$ and $t \leq |x|^c$.
- Proof generation:
  - Let $(P_1, \ldots, P_\ell)$ be the (honest) MIP prover(s).
    Compute ℓ homomorphic evaluations $(\hat{\mathsf{c}}_1, \ldots, \hat{\mathsf{c}}_\ell) := (\mathsf{Eval}(P_1, \mathsf{c}_1), \ldots, \mathsf{Eval}(P_\ell, \mathsf{c}_\ell))$.
  - Output $\pi := \vec{\hat{\mathsf{c}}}$.

Proof verification by $\mathcal{V}$.

- Input: $\left(1^k, \tau, y, \pi\right)$, where $y = (M, x, t)$ and $\pi = \vec{\hat{\mathsf{c}}}$.
- Proof verification:
  - Verify that $t \leq |x|^c$.[a]
  - Decrypt the MIP answers $(a_1, \ldots, a_\ell) := (\mathsf{Dec}^{\mathsf{c}_1}_{\mathsf{sk}_1}(\hat{\mathsf{c}}_1), \ldots, \mathsf{Dec}^{\mathsf{c}_\ell}_{\mathsf{sk}_\ell}(\hat{\mathsf{c}}_\ell))$.
  - If $a_i = \perp$ for some $i \in [\ell]$, reject.
  - Apply the MIP verifier $V(\vec{q}, y, \vec{a})$ to verify the resulting answers.
  - Accept if and only if V accepts.

[a] This is the single place where the verification algorithm depends on c.

Figure 6: Constructing a SNARK for the relation $\mathcal{R}_c$ from a one-round succinct MIP of knowledge and a HEE scheme.

The completeness of the construction follows directly from the completeness of the MIP and the correctness of the HEE scheme. Next, we discuss the succinctness properties of the construction. In the next subsection (Section 10.2), we give a security reduction establishing the (adaptive) proof of knowledge property.

Succinctness. The succinctness of the SNARK easily follows from the succinctness property of the MIP verifier (see Item 3 in Definition 5.2) and the HEE scheme (see Item 4 in Definition 9.1). Indeed:

- the SNARK generator $\mathcal{G}$ uses the encryption algorithm of the HEE scheme to encrypt queries generated by the MIP generator G;
- the SNARK verifier $\mathcal{V}$ uses the decryption algorithm of the HEE scheme to decrypt the answers from the SNARK prover, and runs the MIP verifier;
- the SNARK prover $\mathcal{P}$ uses the evaluation algorithm of the HEE scheme to evaluate each MIP honest prover.

Furthermore, given that (a) the MIP provers can be reduced to circuits that can be computed gate-by-gate in time $\tilde{O}(t)$ and space $s \cdot \mathrm{polylog}(t)$, where t and s are the time and space complexities of $M(x,w)$; (b) the evaluation algorithm of the HEE is local. the SNARK will be complexity-preserving.

10.2 Proof Of Security

We now turn to the detailed proof, which concentrates on constructing and establishing the correctness of a knowledge extractor.

Proposition 10.2 (adaptive proof of knowledge)**.** *For any poly-size $\mathcal{P}^*$ there exists a poly-size extractor $\mathcal{E}_{\mathcal{P}^*}$ such that for all large enough $k \in \mathbb{N}$ and any auxiliary input $z \in \{0,1\}^{\mathrm{poly}(k)}$:*

$$\Pr_{(\vec{q},\vec{\mathsf{sk}},\vec{\mathsf{c}})\leftarrow\mathcal{G}(1^k)}\left[\begin{array}{c|c} \mathcal{V}\big((1^k,\vec{\mathsf{sk}},\vec{q}),y,\vec{\hat{\mathsf{c}}}\big)=1 & (y,\vec{\hat{\mathsf{c}}})\leftarrow\mathcal{P}^*(z,\vec{\mathsf{c}}) \\ w\notin\mathcal{R}_c(y) & w\leftarrow\mathcal{E}_{\mathcal{P}^*}(z,\vec{\mathsf{c}}) \end{array}\right] \leq \mathrm{negl}(k) \ ,$$

where $\vec{q}$ are the MIP queries, $\vec{\mathsf{sk}}$ are the HEE scheme secret keys, and $\vec{\mathsf{c}}$ are the encrypted queries, all generated by $\mathcal{G}$, and $\vec{\hat{\mathsf{c}}}$ are the ciphers generated by $\mathcal{P}^*$ (which are allegedly the result of homomorphically evaluating the MIP provers).

We now describe how the extraction circuit family is constructed and then prove that it satisfies Proposition 10.2. In order to simplify notation, we will address provers $\mathcal{P}^*$ that get as input only $\vec{\mathsf{c}}$; the analysis can be extended to the case where the prover $\mathcal{P}^*$ also gets additional an auxiliary input z drawn from some distribution $\mathcal{Z}$, provided that the underlying HEE scheme is secure with respect to auxiliary inputs drawn from $\mathcal{Z}$. (See discussion in Section 9.1.) We note that, formally, a prover $\mathcal{P}^*$ is a family $\{\mathcal{P}^*_k\}_{k\in\mathbb{N}}$ of poly-size circuits, and so is its corresponding extractor $\mathcal{E}_{\mathcal{P}^*}$; for notational convenience, we omit the subscript k.

The extraction procedure relies on the extraction guarantees of the underlying MIP and HEE schemes. Specifically, for a malicious prover $\mathcal{P}^*$ let $\mathcal{E}^{\mathsf{E}}_{\mathcal{P}^*}$ be its extractor with respect to the HEE scheme E. Given input $\vec{\hat{\mathsf{c}}}$, we first invoke $\mathcal{P}^*(\vec{\mathsf{c}})$ to obtain an instance $y=(M,x,t)$; in case $t > |x|^c$, the extractor aborts. Otherwise, we apply $\mathcal{E}^{\mathsf{E}}_{\mathcal{P}^*}(\vec{\hat{\mathsf{c}}})$ to obtain $\ell(k)$ circuits $\mathsf{C}_1,\ldots,\mathsf{C}_\ell$, which we will treat as defining an MIP prover $P^* := (\mathsf{C}_1,\ldots,\mathsf{C}_\ell)$. We then apply the MIP extractor $E^{P^*}(y)$ to obtain a witness w for y.

We now proceed to prove that (except with negligible probability), whenever $\mathcal{V}$ is convinced by the prover $\mathcal{P}^*$ on an instance of his choosing, the extractor $\mathcal{E}_{\mathcal{P}^*}$ outputs a valid witness for it. The proof is divided into two steps.

In the first step, using the extraction guarantee of the HEE scheme, we deduce that whenever the verifier is convinced, we extract (except with negligible probability) an MIP prover P^* that satisfies the queries $\vec{q}$ encrypted within $\vec{\mathsf{c}}$.

Claim 10.3 (local consistency)**.** *Let $\mathcal{P}^*$ be a poly-size prover and let $\mathcal{E}^{\mathsf{E}}_{\mathcal{P}^*}$ be its HEE extractor. Then for all large enough $k \in \mathbb{N}$,*

$$\Pr_{(\vec{q},\vec{\mathsf{sk}},\vec{\mathsf{c}})\leftarrow\mathcal{G}(1^k)}\left[\begin{array}{c|c} \mathcal{V}((1^k,\vec{\mathsf{sk}},\vec{q}),y,\vec{\hat{\mathsf{c}}})=1 & (y,\vec{\hat{\mathsf{c}}})\leftarrow\mathcal{P}^*(\vec{\mathsf{c}}) \\ V\big(\vec{q},y,(\mathsf{C}_1(q_1),\ldots,\mathsf{C}_\ell(q_\ell))\big)\neq 1 & (\mathsf{C}_1,\ldots,\mathsf{C}_\ell)\leftarrow\mathcal{E}^{\mathsf{E}}_{\mathcal{P}^*}(\vec{\mathsf{c}}) \end{array}\right] \leq \mathrm{negl}(k)\ ,$$

where $\vec{q}$ are the MIP queries, $\vec{\mathsf{sk}}$ are the HEE scheme secret keys, and $\vec{\mathsf{c}}$ are the encrypted queries, all generated by $\mathcal{G}$, and $\vec{\hat{\mathsf{c}}}$ are the (evaluated) ciphers generated by $\mathcal{P}^*$.

Proof. Whenever $\mathcal{V}$ is convinced, the MIP verifier V accepts the decrypted answers induced by the evaluated ciphers $\vec{\hat{\mathsf{c}}}$, i.e., $V\big(\vec{q},y,\mathsf{Dec}^{\mathsf{c}_1}_{\mathsf{sk}_1}(\hat{\mathsf{c}}_1),\ldots,\mathsf{Dec}^{\mathsf{c}_\ell}_{\mathsf{sk}_\ell}(\hat{\mathsf{c}}_\ell)\big)=1$. Furthermore, except with negligible probability over $(\vec{\mathsf{sk}},\vec{\mathsf{c}})$, the HEE extractor $\mathcal{E}^{\mathsf{E}}_{\mathcal{P}^*}$, on input $\vec{\mathsf{c}}$, outputs $\vec{\mathsf{C}}$ such that, for all $i\in[\ell]$, $\mathsf{Dec}_{\mathsf{sk}_i}(\hat{\mathsf{c}}_i)=\mathsf{C}_i(q_i)$. □

Next, in the second step of the proof of Proposition 10.2, we show that if the aforementioned extractor $\mathcal{E}^{\mathsf{E}}_{\mathcal{P}^*}$ outputs an MIP prover $P^*=(\mathsf{C}_1,\ldots,\mathsf{C}_\ell)$ that convinces the MIP verifier with respect to the encrypted queries, then the same prover P^* is sufficiently satisfying for MIP witness extraction; otherwise, the original SNARK prover $\mathcal{P}^*$ and its HEE extractor $\mathcal{E}^{\mathsf{E}}_{\mathcal{P}^*}$ can be used to break the semantic security of the HEE encryption scheme.

Claim 10.4 (from local consistency to extraction)**.** *Let $\mathcal{P}^*$ be a poly-size prover and let $\mathcal{E}_{\mathcal{P}^*}$ be its poly-size SNARK extractor as defined above. Then for all large enough $k\in\mathbb{N}$,*

$$\Pr_{(\vec{q},\vec{\mathsf{sk}},\vec{\mathsf{c}})\leftarrow\mathcal{G}(1^k)}\left[\begin{array}{c|c} t\leq|x|^c & (y,\vec{\hat{\mathsf{c}}})\leftarrow\mathcal{P}^*(\vec{\mathsf{c}}) \\ V\big(\vec{q},y,(\mathsf{C}_1(q_1),\ldots,\mathsf{C}_\ell(q_\ell))\big)=1 \ \wedge & (\mathsf{C}_1,\ldots,\mathsf{C}_\ell)\leftarrow\mathcal{E}_{\mathcal{P}^*}(1^k,\vec{\mathsf{c}}) \\ E^{\mathsf{C}_1,\ldots,\mathsf{C}_\ell}(y)\notin\mathcal{R}_c(y) & y=(M,x,t) \end{array}\right] \leq \mathrm{negl}(k)\ ,$$

where $\vec{q}$ are the MIP queries, $\vec{\mathsf{sk}}$ are the HEE scheme secret keys, and $\vec{\mathsf{c}}$ are the encrypted queries, all generated by $\mathcal{G}$; $\vec{\hat{\mathsf{c}}}$ are the (evaluated) ciphers and y is the instance, both generated by $\mathcal{P}^*$; $\vec{\hat{\mathsf{c}}}$ are the circuits extracted by the SNARK extractor (treated as MIP provers).

Proof. Denote the above event by Bad and assume towards contradiction that, for infinitely many $k\in\mathbb{N}$, Bad occurs with noticeable probability $\delta=\delta(k)$. We show how to break the semantic security of the underlying HEE scheme E.

To break semantic security, we consider the following chosen-plaintext attack (CPA) game:

1. the breaker $\mathcal{B}$ hands its challenger two independent MIP query vectors $(\vec{q}^{\,0},\vec{q}^{\,1})$ sampled using G, and gets back $\vec{\mathsf{c}}_b:=(\mathsf{Enc}_{\mathsf{sk}_1}(q_1^b),\ldots,\mathsf{Enc}_{\mathsf{sk}_\ell}(q_\ell^b))$ for a random $b\in\{0,1\}$ and secret keys $\vec{\mathsf{sk}}$ sampled by the challenger from $\mathsf{Gen}(1^k)$;

2. $\mathcal{B}$ runs the prover $\mathcal{P}^*(\vec{\mathsf{c}}_b)$ and extractor $\mathcal{E}_{\mathcal{P}^*}(\vec{\mathsf{c}}_b)$ to respectively obtain an instance $y=(M,x,t)$ and MIP provers $P^*:=(\mathsf{C}_1,\ldots,\mathsf{C}_\ell)$;

3. $\mathcal{B}$ runs the MIP extractor E, on input y and with oracle access to the extracted MIP prover P^*, to obtain a string $\tilde{w}$ and then verifies whether it is a valid witness for y (which can be done in $\mathrm{poly}(|x|)=\mathrm{poly}(k)$ time); and

4. in case the witness $\tilde{w}$ is valid or $V(\vec{q}^{\,0}, y, P^*(\vec{q}^{\,0})) = V(\vec{q}^{\,1}, y, P^*(\vec{q}^{\,1}))$, $\mathcal{B}$ outputs a random guess for the bit b; otherwise, $\mathcal{B}$ outputs the single b' such that $V(\vec{q}^{\,b'}, y, P^*(\vec{q}^{\,b'})) = 1$.

We now analyze the success probability of $\mathcal{B}$. We define two events X and Y over a random choice of $(\vec{\mathsf{sk}}, \vec{q}^{\,0}, \vec{q}^{\,1})$;[20] note that any choice of $(\vec{\mathsf{sk}}, \vec{q}^{\,0}, \vec{q}^{\,1}, b)$ induces a choice of $y = (M, x, t)$ and P^*. Define X to be the event that $t \leq |x|^c$ and $E^{P^*}(y)$ fails to output a valid witness $\tilde{w}$; next, define Y to be the event that $V(\vec{q}^{\,0}, y, P^*(\vec{q}^{\,0})) \neq V(\vec{q}^{\,1}, y, P^*(\vec{q}^{\,1}))$. Note that:

- by our assumption that Bad occurs with probability δ, we know that

$$\Pr\left[\, V(\vec{q}^{\,b}, y, P^*(\vec{q}^{\,b})) = 1 \;\middle|\; X \,\right] = \frac{\delta}{\Pr[X]} \;\; ;$$

- since E fails at extracting a valid witness from P^* and $\vec{q}^{\,1-b}$ are random MIP queries generated independently of (P^*, y), it must be that

$$\Pr\left[\, V(\vec{q}^{\,1-b}, y, P^*(\vec{q}^{\,1-b})) = 1 \;\middle|\; X \,\right] \leq \varepsilon \leq \mathrm{negl}(k) \;\; .^{21} \tag{9}$$

Combining these two facts, we deduce that

$$\begin{aligned}
&\Pr\left[\, Y \;\middle|\; X \,\right] \\
&\geq \Pr\left[\, V(\vec{q}^{\,b}, y, P^*(\vec{q}^{\,b})) = 1 \wedge V(\vec{q}^{\,1-b}, y, P^*(\vec{q}^{\,1-b})) = 0 \;\middle|\; X \,\right] \\
&\geq \Pr\left[\, V(\vec{q}^{\,b}, y, P^*(\vec{q}^{\,b})) = 1 \;\middle|\; X \,\right] - \Pr\left[\, V(\vec{q}^{\,1-b}, y, P^*(\vec{q}^{\,1-b})) = 1 \;\middle|\; X \,\right] \\
&\geq \frac{\delta}{\Pr[X]} - \mathrm{negl}(k) \;\; ;
\end{aligned} \tag{10}$$

in particular, we can also deduce that

$$\Pr\left[\, X \wedge Y \,\right] \geq \delta - \mathrm{negl}(k) \;\; . \tag{11}$$

By Equations 9 and 10, we have

$$\begin{aligned}
&\Pr\left[\, \mathcal{B} \text{ guesses } b \;\middle|\; X \wedge Y \,\right] \\
&\geq 1 - \Pr\left[\, V(\vec{q}^{\,1-b}, y, P^*(\vec{q}^{\,1-b})) = 1 \;\middle|\; X \wedge Y \,\right] \\
&= 1 - \frac{\Pr\left[\, V(\vec{q}^{\,1-b}, y, P^*(\vec{q}^{\,1-b})) = 1 \wedge Y \;\middle|\; X \,\right]}{\Pr\left[\, Y \;\middle|\; X \,\right]} \\
&\geq 1 - \frac{\mathrm{negl}(k)}{\delta / \Pr\left[\, X \,\right]} \\
&\geq 1 - \mathrm{negl}(k) \;\; .
\end{aligned} \tag{12}$$

[20] We assume without loss of generality that all the coins used for encryption are embedded within the secret keys.

[21] Recall that ε is the knowledge extraction threshold of the MIP. (See Definition 5.2.)

We now deduce that the breaker $\mathcal{B}$ guesses b with a noticeable advantage; indeed, by Equations 11 and 12

$$\begin{aligned}
\Pr\left[\,\mathcal{B}\text{ guesses }b\,\right] &= \Pr\left[\,X\wedge Y\,\right]\Pr\left[\,\mathcal{B}\text{ guesses }b\;\middle|\;X\wedge Y\,\right] + \left(1-\Pr\left[\,X\wedge Y\,\right]\right)\Pr\left[\,\mathcal{B}\text{ guesses }b\;\middle|\;\bar{X}\vee\bar{Y}\,\right] \\
&= \Pr\left[\,X\wedge Y\,\right]\Pr\left[\,\mathcal{B}\text{ guesses }b\;\middle|\;X\wedge Y\,\right] + \left(1-\Pr\left[\,X\wedge Y\,\right]\right)\cdot\frac{1}{2} \\
&= \frac{1}{2} + \Pr\left[\,X\wedge Y\,\right]\left(\Pr\left[\,\mathcal{B}\text{ guesses }b\;\middle|\;X\wedge Y\,\right] - \frac{1}{2}\right) \\
&\geq \frac{1}{2} + \left(\delta-\mathrm{negl}(k)\right)\left(\frac{1}{2}-\mathrm{negl}(k)\right) \\
&\geq \frac{1}{2} + \frac{\delta-\mathrm{negl}(k)}{2}\ ,
\end{aligned}$$

thus completing the proof of the claim. □

Putting it all together. By Claim 10.3 we conclude that (almost) whenever the verifier accepts, $\mathcal{E}_{\mathcal{P}^*}$ extracts a sequence of circuits representing an MIP prover P^* that (locally) satisfies the MIP verifier on the encrypted queries. By Claim 10.4, we deduce that whenever this occurs, P^* must satisfy sufficiently many queries for MIP witness-extraction. This completes the proof of Proposition 10.2 and thus of Theorem 10.1.

10.3 From SNARKs To HEE Schemes

We next show that HEE schemes are in fact not only a sufficient assumption, but also a necessary one (modulo assuming the existence of homomorphic encryption), for the existence of SNARKs.

Proposition 10.5. *If there exist SNARKs, then any $\mathcal{F}$-homomorphic (semantically-secure) encryption scheme $\mathsf{E}=(\mathsf{Gen},\mathsf{Enc},\mathsf{Dec},\mathsf{Eval})$ can be transformed into a new encryption scheme $\tilde{\mathsf{E}}$ that has the homomorphism-extractable property.*

Proof sketch. We concentrate on the case of public key schemes; an analogous result can also be shown for the secret key case.

Let E be an $\mathcal{F}$-homomorphic (public-key) semantically-secure encryption scheme. We construct $\tilde{\mathsf{E}}$ as follows:

- The new generation algorithm $\tilde{\mathsf{Gen}}$, given input 1^k, first runs $\mathsf{Gen}(1^k)$ to obtain secret and a public encryption keys $(\mathsf{sk},\mathsf{pk})$ and then runs the SNARK generator $\mathcal{G}(1^k)$ to obtain a reference string and a private verification state (σ,τ); the new secret and public keys are $(\tilde{\mathsf{sk}},\tilde{\mathsf{pk}}) := ((\mathsf{sk},\tau),(\mathsf{pk},\sigma))$.
- The new encryption algorithm $\tilde{\mathsf{Enc}}$ is the same as the original Enc.
- The new homomorphic evaluation procedure $\tilde{\mathsf{Eval}}$ is defined as follows: given a ciphertext c and a function $f\in\mathcal{F}$, run the original algorithm $\mathsf{Eval}_{\mathsf{pk}}(f,\mathsf{c})$ to obtain an evaluated ciphertext $\hat{\mathsf{c}}$. Then, using the reference string σ compute a SNARK proof π for the statement:

$$y(\mathsf{c},\hat{\mathsf{c}}) := \text{“there is } f \text{ such that } \hat{\mathsf{c}} = \mathsf{Eval}_{\mathsf{pk}}(f,\mathsf{c})\text{”}\ .$$

- The decryption procedure $\tilde{\mathsf{Dec}}^{\mathsf{c}}_{\tilde{\mathsf{sk}}}(\hat{\mathsf{c}})$, on input an (allegedly evaluated) ciphertext $\hat{\mathsf{c}}$ and an original reference ciphertext c, first verifies the SNARK statement $y(\mathsf{c},\hat{\mathsf{c}})$ and, if it verifies, runs the original decryption algorithm $\mathsf{Dec}_{\mathsf{sk}}(\hat{\mathsf{c}})$.

The semantic security of $\tilde{\mathsf{E}}$ follows directly from the semantic security of E. The succinctness of $\tilde{\mathsf{E}}$ follows from the succinctness of E and of the SNARK; indeed the verification time of the SNARK proof π is $p(|\mathsf{c}| + |\hat{\mathsf{c}}| + k)$ for some fixed polynomial p independent of the evaluated function f. The homomorphism-extractable property follows from the (adaptive) proof of knowledge of the SNARK: for any poly-size adversary $\mathcal{A}$ that is given a tuple of ciphers $(\mathsf{c}_1, \ldots, \mathsf{c}_\ell) := (\mathsf{Enc}_{\mathsf{pk}_1}(m_1), \ldots, \mathsf{Enc}_{\mathsf{pk}_\ell}(m_\ell))$ and outputs evaluated ciphers, along with corresponding SNARK proofs, $((\hat{\mathsf{c}}_1, \pi_1), \ldots, (\hat{\mathsf{c}}_\ell, \pi_\ell))$, we define the HEE extractor $\mathcal{E}^{\tilde{\mathsf{E}}}$ to be be the SNARK extractor $\mathcal{E}_{\mathcal{A}}$. (More precisely, we use a SNARK extractor for a multiple-statement adversary, as given by Lemma 5.19.) By the SNARK proof of knowledge guarantee, (except with negligible probability) whenever any of $\mathcal{A}$'s outputs $(\hat{\mathsf{c}}_i, \pi_i)$ passes SNARK verification, the SNARK extractor outputs a corresponding witness f_i such that $\mathsf{Eval}_{\mathsf{pk}}(f_i, \mathsf{c}_i) = \hat{\mathsf{c}}_i$, and hence such that $f_i(m_i) = \mathsf{Dec}_{\mathsf{sk}}(\hat{\mathsf{c}}_i)$. □

Remark 10.6. We note that:

1. If one is only interested in scenarios with a *single* evaluator (for example, this is the case in our application of constructing SNARKs), then HEE schemes with a *single* secret and public key that will respectively be used to jointly encrypt and decrypt plaintexts and ciphertexts will likely suffice. In such a case, we could still be able to construct such HEE schemes form SNARKs, by generating a single SNARK proof ensuring good evaluation of each ciphertext, rather than a SNARK proof for each ciphertext.

2. Given a SNARK we can of course construct encryption schemes with extractability notions that are stronger than mere homomorphism extractability; e.g., we could also enforce that the evaluated function is taken from a prescribed function family $\mathcal{F}$ (which can be seen as a "one-hop" targeted-malleability property [BSW12]).

Acknowledgements

We thank Eli Ben-Sasson for discussions about MIP constructions. We thank Ran Canetti, Omer Paneth, and Ben Riva for valuable discussions on MIP-based SNARKs. We also thank Ben for referring us to [CL10].

References

[ABOR00] William Aiello, Sandeep N. Bhatt, Rafail Ostrovsky, and Sivaramakrishnan Rajagopalan. Fast verification of any remote procedure call: Short witness-indistinguishable one-round proofs for NP. In *Proceedings of the 27th International Colloquium on Automata, Languages and Programming*, ICALP '00, pages 463–474, 2000.

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998. Preliminary version in FOCS '92.

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: a new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998. Preliminary version in FOCS '92.

[AV77] Dana Angluin and Leslie G. Valiant. Fast probabilistic algorithms for hamiltonian circuits and matchings. In *Proceedings on 9th Annual ACM Symposium on Theory of Computing*, STOC '77, pages 30–41, 1977.

[Bab85] László Babai. Trading group theory for randomness. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*, STOC '85, pages 421–429, 1985.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, 1988.

[BCCT12a] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 326–349, 2012.

[BCCT12b] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for SNARKs and proof-carrying data. Cryptology ePrint Archive, Report 2012/095, 2012.

[BCI+13] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In *Proceedings of the 10th Theory of Cryptography Conference*, TCC '13, pages ???–???, 2013.

[BFL90] László Babai, Lance Fortnow, and Carsten Lund. Nondeterministic exponential time has two-prover interactive protocols. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, SFCS '90, pages 16–25, 1990.

[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*, STOC '91, pages 21–32, 1991.

[BG08] Boaz Barak and Oded Goldreich. Universal arguments and their applications. *SIAM Journal on Computing*, 38(5):1661–1694, 2008. Preliminary version appeared in CCC '02.

[BGV11] Siavosh Benabbas, Rosario Gennaro, and Yevgeniy Vahlis. Verifiable delegation of computation over large datasets. In *Proceedings of the 31st Annual International Cryptology Conference*, CRYPTO '11, pages 111–131, 2011.

[BHZ87] Ravi B. Boppana, Johan Håstad, and Stathis Zachos. Does co-NP have short interactive proofs? *Information Processing Letters*, 25(2):127–132, 1987.

[BK89] Manuel Blum and Sampath Kannan. Designing programs that check their work. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, STOC '89, pages 86–97, 1989.

[BOGKW88] Michael Ben-Or, Shafi Goldwasser, Joe Kilian, and Avi Wigderson. Multi-prover interactive proofs: how to remove intractability assumptions. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, STOC '88, pages 113–131, 1988.

[BP04] Mihir Bellare and Adriana Palacio. Towards plaintext-aware public-key encryption without random oracles. In *Proceedings of the 10th International Conference on the Theory and Application of Cryptology and Information Security*, ASIACRYPT '04, pages 48–62, 2004.

[BSCGT12] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, and Eran Tromer. On the concrete-efficiency threshold of probabilistically-checkable proofs, 2012. Electronic Colloquium on Computational Complexity, TR12-045.

[BSCGT13] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, and Eran Tromer. Fast reductions from RAMs to delegatable succinct constraint satisfaction problems. In *Proceedings of the 4th Innovations in Theoretical Computer Science Conference*, ITCS '13, pages ???–???, 2013.

[BSGH+05] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. Short PCPs verifiable in polylogarithmic time. In *Proceedings of the 20th Annual IEEE Conference on Computational Complexity*, CCC '05, pages 120–134, 2005.

[BSS08] Eli Ben-Sasson and Madhu Sudan. Short PCPs with polylog query complexity. *SIAM Journal on Computing*, 38(2):551–607, 2008.

[BSW12] Dan Boneh, Gil Segev, and Brent Waters. Targeted malleability: Homomorphic encryption for restricted computations. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 350–366, 2012.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science*, FOCS '11, 2011.

[CHS05] Ran Canetti, Shai Halevi, and Michael Steiner. Hardness amplification of weakly verifiable puzzles. In *Proceedings of the 2nd Theory of Cryptography Conference*, TCC '05, pages 17–33, 2005.

[CKV10] Kai-Min Chung, Yael Kalai, and Salil Vadhan. Improved delegation of computation using fully homomorphic encryption. In *Proceedings of the 30th Annual International Cryptology Conference*, CRYPTO '10, pages 483–501, 2010.

[CL10] Kai-Min Chung and Feng-Hao Liu. Parallel repetition theorems for interactive arguments. In *Proceedings of the 7th Theory of Cryptography Conference*, TCC '10, pages 19–36, 2010.

[CR72] Stephen A. Cook and Robert A. Reckhow. Time-bounded random access machines. In *Proceedings of the 4th Annual ACM Symposium on Theory of Computing*, STOC '72, pages 73–80, 1972.

[CT10] Alessandro Chiesa and Eran Tromer. Proof-carrying data and hearsay arguments from signature cards. In *Proceedings of the 1st Symposium on Innovations in Computer Science*, ICS '10, pages 310–331, 2010.

[DCL08] Giovanni Di Crescenzo and Helger Lipmaa. Succinct NP proofs from an extractability assumption. In *Proceedings of the 4th Conference on Computability in Europe*, CiE '08, pages 175–185, 2008.

[DFH12] Ivan Damgård, Sebastian Faust, and Carmit Hazay. Secure two-party computation with low communication. In *Proceedings of the 9th Theory of Cryptography Conference*, TCC '12, pages 54–74, 2012.

[DLN+04] Cynthia Dwork, Michael Langberg, Moni Naor, Kobbi Nissim, and Omer Reingold. Succinct NP proofs and spooky interactions, December 2004. Available at `www.openu.ac.il/home/mikel/papers/spooky.ps`.

[FG12] Dario Fiore and Rosario Gennaro. Publicly verifiable delegation of large polynomials and matrix computations, with applications. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 501–512, 2012.

[FGL+91] Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. Approximating clique is almost NP-complete (preliminary version). In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science*, SFCS '91, pages 2–12, 1991.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Proceedings of the 6th Annual International Cryptology Conference*, CRYPTO '87, pages 186–194, 1987.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, STOC '09, pages 169–178, 2009.

[GGP10] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: outsourcing computation to untrusted workers. In *Proceedings of the 30th Annual International Cryptology Conference*, CRYPTO '10, pages 465–482, 2010.

[GGPR12] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. Cryptology ePrint Archive, Report 2012/215, 2012.

[GH98] Oded Goldreich and Johan Håstad. On the complexity of interactive proofs with bounded communication. *Information Processing Letters*, 67(4):205–214, 1998.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: Interactive proofs for Muggles. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, STOC '08, pages 113–122, 2008.

[GLR11] Shafi Goldwasser, Huijia Lin, and Aviad Rubinstein. Delegation of computation without rejection problem from designated verifier CS-proofs. Cryptology ePrint Archive, Report 2011/456, 2011.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989. Preliminary version appeared in STOC '85.

[Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In *Proceedings of the 16th International Conference on the Theory and Application of Cryptology and Information Security*, ASIACRYPT '10, pages 321–340, 2010.

[GVW02] Oded Goldreich, Salil Vadhan, and Avi Wigderson. On interactive proofs with a laconic prover. *Computational Complexity*, 11(1/2):1–53, 2002.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing*, STOC '11, pages 99–108, 2011.

[Hai09] Iftach Haitner. A parallel repetition theorem for any interactive argument. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '09, pages 241–250, 2009.

[Har04] Prahladh Harsha. *Robust PCPs of Proximity and Shorter PCPs*. PhD thesis, MIT, EECS, September 2004.

[HPWP10] Johan Håstad, Rafael Pass, Douglas Wikström, and Krzysztof Pietrzak. An efficient parallel repetition theorem. In *TCC '10*, pages 1–18, 2010.

[IKO07] Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. Efficient arguments without short PCPs. In *Proceedings of the Twenty-Second Annual IEEE Conference on Computational Complexity*, CCC '07, pages 278–291, 2007.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, STOC '92, pages 723–732, 1992.

[KR06] Yael Tauman Kalai and Ran Raz. Succinct non-interactive zero-knowledge proofs with preprocessing for LOGSNP. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 355–366, 2006.

[KR09] Yael Tauman Kalai and Ran Raz. Probabilistically checkable arguments. In *Proceedings of the 29th Annual International Cryptology Conference*, CCC '09, pages 143–159, 2009.

[KRR12] Yael Kalai, Ran Raz, and Ron Rothblum. Where delegation meets Einstein. Isaac Newton Institute for Mathematical Sciences, Formal and Computational Cryptographic Proofs, 2012.

[KZG10] Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques*, ASIACRYPT '10, pages 177–194, 2010.

[Lip12] Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In *Proceedings of the 9th Theory of Cryptography Conference*, TCC '12, pages 169–189, 2012.

[Mic00] Silvio Micali. Computationally sound proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2000. Preliminary version appeared in FOCS '94.

[Mie08] Thilo Mie. Polylogarithmic two-round argument systems. *Journal of Mathematical Cryptology*, 2(4):343–363, 2008.

[MR08] Dana Moshkovitz and Ran Raz. Two-query PCP with subconstant error. *Journal of the ACM*, 57:1–29, June 2008. Preliminary version appeared in FOCS '08.

[Nao03] Moni Naor. On cryptographic assumptions and challenges. In *Proceedings of the 23rd Annual International Cryptology Conference*, CRYPTO '03, pages 96–109, 2003.

[PRV12] Bryan Parno, Mariana Raykova, and Vinod Vaikuntanathan. How to delegate and verify in public: Verifiable computation from attribute-based encryption. In *Proceedings of the 9th Theory of Cryptography Conference*, TCC '12, pages 422–439, 2012.

[PST11] Charalampos Papamanthou, Elaine Shi, and Roberto Tamassia. Signatures of correct computation. Cryptology ePrint Archive, Report 2011/587, 2011.

[Rot11] Ron Rothblum. Homomorphic encryption: From private-key to public-key. In *Proceedings of the 8th Theory of Cryptography Conference*, TCC '11, pages 219–234, 2011.

[RS97] Ran Raz and Shmuel Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability pcp characterization of np. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, STOC '97, pages 475–484, 1997.

[RV09] Guy N. Rothblum and Salil Vadhan. Are PCPs inherent in efficient arguments? In *Proceedings of the 24th IEEE Annual Conference on Computational Complexity*, CCC '09, pages 81–92, 2009.

[Sha92] Adi Shamir. IP = PSPACE. *Journal of the ACM*, 39(4):869–877, 1992.

[TS96] Amnon Ta-Shma. A note on PCP vs. MIP. *Information Processing Letters*, 58:135–140, May 1996.

[Val08] Paul Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In *Proceedings of the 5th Theory of Cryptography Conference*, TCC '08, pages 1–18, 2008.

[vDGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *Proceedings of the 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, EUROCRYPT '10, pages 24–43, 2010.

[Wee05] Hoeteck Wee. On round-efficient argument systems. In *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming*, ICALP '05, pages 140–152, 2005.