Efficient Arguments without Short PCPs*

Yuval Ishai†
Computer Science Department, Technion
yuvali@cs.technion.ac.il

EyalKushilevitz‡
Computer Science Department, Technion
eyalk@cs.technion.ac.il

Rafail Ostrovsky§
CS and Math Departments, UCLA
rafail@cs.ucla.edu

IEEE Conference on Computational Complexity 2007: 278-291

Abstract

Current constructions of efficient argument systems combine a short (polynomial size) PCP with a cryptographic hashing technique. We suggest an alternative approach for this problem that allows to simplify the underlying PCP machinery using a stronger cryptographic technique.

More concretely, we present a direct method for compiling an exponentially long PCP which is succinctly described by a *linear* oracle function $\pi : F^n \to F$ into an argument system in which the verifier sends to the prover $O(n)$ encrypted field elements and receives $O(1)$ encryptions in return. This compiler can be based on an arbitrary homomorphic encryption scheme. Applying our general compiler to the exponential size Hadamard code based PCP of Arora et al. (JACM 1998) yields a simple argument system for NP in which the communication from the prover to the verifier only includes a constant number of short encryptions.

The main tool we use is a new cryptographic primitive which allows to efficiently commit to a linear function and later open the output of the function on an arbitrary vector. Our efficient implementation of this primitive is independently motivated by cryptographic applications.

1 Introduction

We revisit the problem of constructing *efficient argument systems* for NP — computationally sound interactive proofs in which the amounts of communication and computation performed by the verifier are smaller than in classical proofs. Solutions for this fundamental problem have been among the greatest meeting points of complexity theory and cryptography, combining sophisticated PCP machinery with a clever cryptographic hashing technique. In this work we suggest a *different*, more *general*, and arguably *simpler*

*Work done in part while the authors were visiting IPAM.

†Research supported by grant 1310/06 from the Israel Science Foundation and grant 2004361 from the U.S.-Israel Binational Science Foundation.

‡Research supported by grant 1310/06 from the Israel Science Foundation and grant 2002354 from the U.S.-Israel Binational Science Foundation.

§Research supported by grant 2002354 from the U.S.-Israel Binational Science Foundation, IBM Faculty Award, Xerox Innovation Group Award, NSF Cybertrust grant no. 0430254, and U.C. MICRO grant.

approach for tackling this problem by relying on stronger cryptographic machinery. Before describing our approach, we give some background to put it in context.

1.1 Background

Classical proof systems for NP languages require the prover to send the entire witness to the verifier. In particular, if $L \in \mathrm{NTIME}(T(n))$ then proving that $x \in L$ generally requires the prover to send a message of length $\Omega(T(n))$ and the verifier to spend $\Omega(T(n))$ time verifying the proof. Moreover, classical proofs cannot be generally used to speed up the verification of a *deterministic* polynomial-time computation, a goal which is strongly motivated by real life applications. It is thus natural to ask whether it is possible to avoid these limitations, possibly by allowing randomness and interaction.

Interactive proof systems, as defined by Goldwasser, Micali, and Rackoff [21], relax classical proofs by allowing randomness and interaction, but still require soundness to hold with respect to computationally *unbounded* provers. Unfortunately, proof systems of this type do not seem to be helpful in the current context either. In particular, the existence of interactive proof systems for NP in which the prover is *laconic*, in the sense that the total length of the messages it sends to the verifier is much smaller than the length of the witness, would imply a nontrivial nondeterministic simulation for coNP [18, 19]. A similar limitation holds for proof systems in which the verifier's time complexity is smaller than the length of the witness.

Efficient arguments. Quite remarkably, it is possible to break the efficiency barriers by relaxing the soundness requirement to hold only with respect to computationally bounded provers. Such interactive proof systems, introduced by Brassard et al. [9], are referred to as *arguments*. Under cryptographic assumptions, Kilian [23] constructed an interactive argument system for NP with a polylogarithmic communication complexity. Micali [26] suggested a *non-interactive* implementation of argument systems, termed CS proofs, whose soundness was proved in the Random Oracle Model. Combined with modern machinery (e.g., [7]), these protocols give rise to "essentially optimal" argument systems in which the communication complexity, computational complexity of the verifier, and computational complexity of the prover are all optimal up to polylogarithmic factors. In addition to their direct motivation, efficient arguments have also found diverse applications in cryptography, cf. [4, 5, 28].

PCPs. All known constructions of efficient argument systems rely on *Probabilistically Checkable Proofs* (PCPs), also known as "holographic proofs". A PCP is a redundant encoding of a classical proof that can be probabilistically verified by querying a small number of bits. The celebrated PCP Theorm [2, 1] asserts that every classical proof can be turned into a polynomial-size PCP which can be verified using a constant number of queries (up to a small constant soundness error). The study of PCPs was initiated by Babai et al. [3] and by Feige et al. [16] with a very different motivation. While in [3] PCPs were used as a means of speeding up verification of proofs and computation, their study in [16], as well as in most subsequent works in the area, was mainly driven by the exciting connection between PCPs and hardness of approximation. The current work is related to the former application of PCPs, which has also motivated a fruitful line of work on obtaining *shorter* PCPs [31, 6, 8, 7, 13]. Despite some recent simplifications of PCP constructions [6, 14], culminating in the elegant new proof of the PCP Theorem by Dinur [13], the construction and analysis of efficient PCPs is still quite far from straightforward.

From PCPs to efficient arguments. Applying PCPs to enable efficient verification "in the real world" is not as straightforward as it may first appear. Consider the following natural attempt to implement an efficient interactive proof based on a PCP. The interactive verifier emulates the PCP verifier, sending to the prover

the (randomized) set of indices it would like to probe; the interactive prover then emulates the PCP prover, responding with the corresponding bits of the proof. An inherent problem with this direct approach is that the soundness of the PCP is only guaranteed if the proof string is *fixed in advance*, independently of the verifier's queries. In the above implementation, the prover can easily violate the soundness of the PCP by appropriately correlating its answers with the queries. Note that we cannot afford to have the prover send the entire PCP string to the verifier, as the length of this string is bigger than the length of the witness.

The key idea that was used in [23] for getting around these difficulties is to first require the prover to send a *succinct commitment* to the PCP string, using a suitable cryptographic technique, and then allow the verifier to open a small, randomly chosen, subset of the committed bits. In order to efficiently support the required functionality, a special type of cryptographic commitment should be used. Specifically, the commitment scheme should be: (1) computationally binding (but not necessarily hiding); (2) *succinct*, in the sense that it involves a small amount of communication between the prover and the verifier; and (3) support *efficient local decommitment*, in the sense that following the commitment phase any bit of the committed string can be opened (or "decommitted") very efficiently, in particular without opening the entire string. An implementation of such a commitment scheme can be based on any collision-resistant hash function by using Merkle's tree hashing technique [25]. Roughly speaking, the commitment process proceeds by first viewing the bits of the proof as leaves of a complete binary tree, and then using the hash function to compute the label of each node from the two labels of its children. The succinct commitment string is taken as the label of the root. To open a selected bit of the proof, the labels of all nodes on a path from the corresponding leaf to the root, along with their siblings, are revealed. Thus, the number of atomic cryptographic operations required to open a single bit of the proof is logarithmic in the proof length.

1.2 Our Contribution

As indicated above, all previous constructions of efficient argument system relied on the following two-stage approach: (1) convert a classical proof into an explicit *polynomial-size* PCP string (an "encoded proof"); (2) apply the tree-based cryptographic hashing technique to commit to the proof, and later open a small set of bits chosen by the verifier.

This state of affairs raises the question of obtaining an alternative approach for constructing efficient argument systems, one that avoids the need for (complicated) polynomial-size PCPs. Our main motivation for considering this question comes from the goal of obtaining conceptually *simpler* argument systems, ones that do not require full PCP machinery. However, as we discuss below, the question is also motivated by efficiency considerations. Note that in Step (1) above a classical proof is expanded by introducing a significant amount of redundancy, whereas in Step (2) it dramatically shrinks in size. This calls for a more direct approach, or a shortcut, that combines the two steps into one.

The idea. Our main idea is that for the purpose of constructing efficient arguments it is not essential for the PCP string to be of polynomial size. Instead, we view the PCP as a *function* $\pi : F^n \to F$, whose domain may be exponentially large, but whose evaluation can be carried out in polynomial time. We refer to π as a *linear PCP* if the honest prover computes a linear function over a finite field F. In this work we will focus on the class of linear PCPs, but the approach can be generalized to other function classes (e.g., using the degree-2 homomorphic encryption of [11]).

A similar view of PCPs as oracles is abundant in the PCP literature. However, PCP oracles of exponential domain size were typically used as intermediate steps towards efficient constructions rather than end results that can be directly plugged into applications. Most notably, an exponential size linear PCP based

on Hadamard codes was used by Arora et al. [1] as the simpler component in the proof of the PCP theorem. The same PCP can be used as the base case in Dinur's recent alternative proof of the PCP theorem [13].

Our results. Our main result is a general compiler, which takes an arbitrary linear PCP $\pi : F^n \to F$ as above, and turns it into an argument system with a very laconic prover. Specifically, in this argument system the prover communicates to the verifier only a constant number of encrypted field elements for each PCP query, whereas the verifier communicates to the prover $O(n)$ encrypted field elements. Our compiler can be based on any *homomorphic encryption scheme* (e.g., the one of Goldwasser and Micali [20], ElGamal [15], or Paillier [30]).[1] Instantiated with the simple linear PCP based on Hadamard codes [1], we get an argument system with an expensive (yet polynomial) communication from the verifier to the prover, but with an extremely short communication from the prover to the verifier which only consists of a constant number of encrypted field elements.

We stress that even in the above setting of laconic provers, with an arbitrary (polynomial) communication from the verifier to the prover, the previous methodology could not yield argument systems without relying on short (polynomial-size) PCPs.

Optimizations. While we view the above general result as our main conceptual contribution, we also suggest several approaches for building on the laconic prover feature of the above concrete protocol to get an overall efficient solution. A first approach is by *batching* multiple proofs: our protocol allows the prover to convince the verifier of an arbitrary (polynomial) number of statements using essentially the same amount of communication from the verifier to the prover as that required for proving a single statement. In practical terms, this means that the verifier can simultaneously verify many long computations at a total cost which is comparable to that of performing a *single* computation on its own. As an additional optimization, we suggest a heuristic approach for shifting almost all of the communication originating from the verifier to an offline phase, which can be carried out before any inputs are available. As before, the cost of this offline phase can be amortized over many different statements. Finally, we sketch an approach which combines a communication balancing technique with *cryptographic* recursion, yielding arguments with a total communication complexity of $O(n^\epsilon)$ and $O(1/\epsilon)$ rounds, for an arbitrary $\epsilon > 0$. The latter approach, however, sacrifices both the simplicity and the potential efficiency advantages of our main protocol over alternative approaches.

Potential efficiency advantages. As noted above, state-of-the-art PCP constructions leave little to be desired in terms of asymptotic efficiency. However, the practical motivation of verifying computations may call for a more refined efficiency analysis. From this point of view, we believe that our approach has potential to yield better efficiency, at least in some circumstances. For instance, the optimized variant of our construction requires the verifier to perform only a constant number of cryptographic operations (following preprocessing), as opposed to a logarithmic number of operation in the traditional approach. In terms of the prover's efficiency, our approach does not inherently require the prover to compute a redundant encoding of its input. This suggests the possibility of designing PCPs that are optimized to make better use of the "implicit encoding" feature of our approach.

Techniques. Our main result relies on a generalized commitment primitive that allows to efficiently commit the prover to an *arbitrary linear function* $\pi : F^n \to F$, and later efficiently open the value of this function on an arbitrary vector. Note that the standard hashing-based commitment primitive can be viewed as a special

[1] Alternatively, our compiler can use any deterministic, homomorphic one-way commitment [22], which can be based on the intractability of discrete logarithms.

case in which only evaluations on unit vectors are allowed. The technique we use to efficiently implement this primitive is as follows. To commit the prover to a linear function π, the verifier encrypts a random input r using a homomorphic encryption scheme, and asks the prover to compute an encryption of $\pi(r)$. (This can be done by the prover based on the homomorphism property, without knowing the secret key.) Then, to decommit the value $\pi(q)$, the verifier asks the prover to reveal $\pi(q)$ together with $\pi(r + \alpha q)$ for a random field element α chosen by the verifier. The second evaluation point is used by the verifier to perform a consistency check. We show that any prover who breaks the binding requirement can be used to break the semantic security of the underlying encryption.

Our new cryptographic primitive is independently motivated by cryptographic applications. For instance, it can be used to efficiently commit to a large statistical database and later open the average (equivalently, sum) of a selected subset of the entries.

Related work. Motivated by the goal of obtaining simpler polynomial-size PCPs, Zimand [33] suggests a way to "derandomize" the Hadamard-based PCP of Arora et al. [1] by settling for computational soundness. A major disadvantage of the approach from [33] is that in order to guarantee soundness against malicious provers which run in time $O(n^t)$, the honest prover is required to run in time $\omega(n^t)$. In contrast, the argument systems we obtain satisfy the traditional notion of computational soundness, that holds with respect to every polynomial-time malicious prover.

Organization. The rest of the paper is organized as follows. Following some definitions (Section 2) in Section 3 we introduce the "commitment with linear decommitment" primitive and efficiently implement it using any homomorphic encryption scheme. In Section 4 we use this commitment primitive to compile linear PCPs into efficient arguments. In Section 5 we give a self-contained presentation of the Hadamard-based linear PCP from [1] in our framework, which can be combined with the compiler from Section 4 to give laconic-prover arguments for NP. Finally, in Section 6 we sketch an approach for transforming the latter argument systems into ones in which the communication complexity is sublinear in both directions.

2 Preliminaries

In this section we define the main proof systems and cryptographic primitives on which we rely. In the following we will let n denote an input length parameter, or alternatively a (polynomially related) proof length parameter. The running time of all honest parties is restricted to be polynomial in n. We will also use additional parameters, such as a cryptographic security parameter k, an underlying field F, and a soundness parameter ϵ; for convenience, these parameters can all be thought of as being determined by n.

2.1 Arguments

Arguments are computationally sound interactive proof systems, as defined by [21, 9]. An argument protocol is defined by a two interactive PPT algorithms: a prover P and a verifier V. (In fact, for our purposes the prover may be deterministic.) An argument protocol (P, V) for an NP language L with soundness error $\epsilon(\cdot)$ should satisfy the following requirements.

Completeness: For every $x \in L$ and corresponding NP witness w, the interaction of $V(x)$ with $P(x, w)$ always makes V accept.

Soundness: For every $x \notin L$ and every *efficient* (but possibly non-uniform) malicious prover P^*, the probability that the interaction of $V(x)$ with $P^*(x)$ leads V to accept is at most $\epsilon(|x|)$, except perhaps for finitely many x.

We consider soundness error $\epsilon(n)$ that is either a constant or a negligible function. While we do not explicitly consider the stronger *proof of knowledge* property of arguments (cf. [5]), our protocols satisfy this property as well.

2.2 Linear PCPs and MIPs

The starting point for our constructions of efficient arguments is a *linear* proof system, in which the honest prover computes linear functions (over some underlying finite field F) of the verifier's queries. Our general compiler of linear PCPs into efficient arguments takes the simplest form when applied to a multi-prover variant of linear PCPs, referred to as *linear MIPs*. A simple standard transformation of an arbitrary linear PCP (in particular, the Hadamard-based PCP from [1]) into a linear MIP will be described in Section 5.

Loosely speaking, a linear MIP (Multiprover Interactive Proof) consists of an ℓ-tuple of proof oracles $(\pi_1, \ldots, \pi_\ell)$, where each π_i is a *linear* function $\pi_i : F^n \to F$ over some finite field F. The verifier V picks an ℓ-tuple of queries $(q_1, \ldots, q_\ell)$, where each q_i is in F^n, and gets in return $\pi_1(q_1), \ldots, \pi_\ell(q_\ell)$; based on the input x and these answers, V either accepts or rejects. The completeness requirement is that for every $x \in L$ there exist linear functions $(\pi_1, \ldots, \pi_\ell)$ as above which make V accept with probability 1. The soundness requirement is that for every $x \notin L$, and any (maliciously chosen, and *possibly non-linear*) proof functions $(\tilde{\pi}_1, \ldots, \tilde{\pi}_\ell)$, the probability that V accepts is at most ϵ. Similarly to PCPs, the crucial feature of MIPs is that the proof functions $\tilde{\pi}_j$ must be fixed *before* the queries are randomly picked by V. When compiling MIPs into arguments, we will need to simulate such independence in an interactive setting which allows the prover to pick its answers after seeing all queries made by the verifier.

More formally, a *linear MIP* for an NP language L consists of a probabilistic polynomial-time verifier V and a polynomial-time prover algorithm P, which will be used to implement multiple provers. Any input $x \in \{0,1\}^*$ determines a proof length parameter $n = \text{poly}(|x|)$, number of provers $\ell = \ell(n)$, a finite field $F = F(n)$, and a soundness error parameter $\epsilon = \epsilon(n)$. These parameters are known to both the verifier and the provers. Once an input x and a corresponding NP witness w are fixed, we let π_i denote $P(i, x, w, \cdot)$ and view P as an ℓ-tuple of proof functions $(\pi_1, \ldots, \pi_\ell)$ returning answers to the verifier's queries q_i. We require that every π_i is a *linear* function $\pi_i : F^n \to F$. (A function π is linear over F if for all $q, q' \in F^n$ we have $\pi(q + q') = \pi(q) + \pi(q')$.) The interaction between V and P proceeds by having V, based on x, pick an ℓ-tuples of queries $(q_1, \ldots, q_\ell)$, where each q_i is in F^n. Every q_i is sent to the corresponding prover, who responds with $\pi_i(q_i)$. Finally, based on its random input and the ℓ answers, the verifier decides whether to accept or reject. A linear MIP with soundness error ϵ should satisfy the following requirements.

Completeness: For every $x \in L$ and corresponding NP witness w, we have

$$\Pr[V(x, q_1, \ldots, q_\ell, \pi_1(q_1), \ldots, \pi_\ell(q_\ell)) = ACC] = 1,$$

where the probability is over the random choice of queries $(q_1, \ldots, q_\ell)$ by V and where $\pi_i(q_i) \stackrel{\text{def}}{=} P(i, x, w, q_i)$.

Soundness: For every $x \notin L$ and (possibly non-linear and computationally inefficient) proof functions $(\tilde{\pi}_1, \ldots, \tilde{\pi}_\ell)$, we have

$$\Pr[V(x, q_1, \ldots, q_\ell, \tilde{\pi}_1(q_1), \ldots, \tilde{\pi}_\ell(q_\ell)) = ACC] \leq \epsilon(n),$$

where again the probability is over the randomness of V.

Choice of parameters. By default, we fix the soundness error $\epsilon(n)$ to be a constant, say 1/2, in which case it will suffice to let $\ell(n) = O(1)$. The soundness error can be decreased to $2^{-\sigma}$ by using σ independent repetitions (with disjoint sets of provers), letting $\ell = O(\sigma)$. It will be convenient to assume that the size of the field $F = F(n)$ grows (slightly) super-polynomially with n, though our constructions can work over $F = \text{GF}(2)$ as well.

2.3 Homomorphic Encryption

Homomorphic encryption is a semantically secure [20] public-key encryption $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ in which plaintexts are taken from a finite group, and the following homomorphism property holds. Given any public key pk generated by Gen and any two valid ciphertexts $c_1 \in \mathsf{Enc}(pk, m_1)$ and $c_2 \in \mathsf{Enc}(pk, m_2)$, it is possible to *efficiently* compute a random encryption $\mathsf{Enc}(pk, m_1 + m_2)$. Thus, one can also efficiently compute $\mathsf{Enc}(pk, \alpha \cdot m_1)$ for a known integer α.

It will be convenient to describe our protocols using a homomorphic encryption scheme over a finite field F. By this we mean that the plaintext group is the additive group of F. We also require that given $\mathsf{Enc}(pk, m)$ and $\alpha \in F$ it is possible to efficiently compute $\mathsf{Enc}(pk, \alpha \cdot m)$. This follows automatically if F is of prime order, and can be obtained for an extension field by viewing it as a linear space over the base field. For instance, one can implement homomorphic encryption over $\text{GF}(2^\sigma)$ from homomorphic encryption over $\text{GF}(2)$ (e.g., the Goldwasser-Micali cryptosystem [20]) by using σ homomorphic encryptions over $\text{GF}(2)$ to implement a single homomorphic encryption over $\text{GF}(2^\sigma)$.

We finally note that our protocols can be generalized to work with homomorphic encryption over general Abelian groups (that are not necessarily of a prime order), or with weaker homomorphic primitives that do not allow efficient decryption and only satisfy "one-way" security (rather than semantic security). All these primitives are known to imply collision-resistant hash-functions [22], which underly the traditional constructions of efficient arguments.

Choice of parameters. The encryption scheme uses a security parameter k, that is given to the key generation algorithm Gen as input, and its security is defined with respect to k. In our higher level protocols, k can be thought of as being determined by the input length parameter n. More concretely, under standard cryptographic assumptions one can let $k(n) = n^c$ for an arbitrarily small constant $c > 0$, and under "exponential strength" cryptographic assumptions one can let $k(n) = \text{polylog } n$.

3 Commitment with Linear Decommitment

In this section we define and efficiently implement a "commitment with linear decommitment" primitive, which we later use to compile MIPs into arguments. In Section 3.1 we define a basic version of this primitive which is implemented in Section 3.2. In Section 3.3 we implement an extended version that supports multiple (parallel) commitments and decommitments.

3.1 Definition

A *commitment with linear decommitment* is a protocol for two parties: a sender S and a receiver R. The protocol consists of two phases. In the *commitment phase*, S has an input $d \in F^n$ representing a linear function $f_d : F^n \to F$ (such that $f_d(q) = \langle d, q \rangle$, the inner product of d, q) and R has no input. The two

parties interact with each other and have no output, though R may keep some "decommitment information" to be used in the next phase. In the *decommitment phase*, R has input q (a "decommitment query") and the decommitment information, and at the end of this phase it should either reject or output a value $a \in F$. If both parties are honest, the output of R should satisfy $a = f_d(q)$. In addition, the protocol should satisfy a computational *binding* property. Loosely speaking, for any efficient malicious sender, the following should hold. After the commitment phase there is a function $\tilde{f}$ (possibly different from f_d and possibly non-linear) such that, for any decommitment query q, the receiver either outputs $\tilde{f}(q)$ or rejects (except with negligible probability).

More formally, the sender and the receiver are defined by a pair of interactive PPT algorithms (S, R). Both S and R use independent random inputs for the two phases of the protocol. To simplify the use of quantifiers in the following definition and its extensions, it will be convenient to think of the inputs of the two phases as being generated by an *environment* $\mathcal{E}$, which given n can produce arbitrary inputs $d, q \in F^n$. Note that $\mathcal{E}$ does not have access to the protocol's transcript; thus, the decommitment query q it generates is independent of the random inputs of S and R. The interaction of S and R with $\mathcal{E}$, on length parameter n, defines the following two-phase experiment.

- *Commitment phase:* $\mathcal{E}$ gives to S inputs F and $d \in F^n$ and to R the inputs n and F. These inputs, along with the random inputs of S and R, determine an interaction between S and R, at the end of which S (resp., R) locally keeps a decommitment information string z_S (resp., z_R) for use in the next phase.

- *Decommitment phase:* $\mathcal{E}$ gives to R a decommitment query $q \in F^n$. This input, along with z_S, z_R and the random inputs of S and R (that are chosen independently of the random inputs in the first phase), determines further interaction between S and R. At the end of this interaction, R either outputs a value $a \in F$ or the symbol $\perp$ ("reject").

The above experiment can now be used to formally define our commitment primitive.

Definition 3.1 *A* commitment with linear deommitment *is defined by a pair of PPT algorithms (S, R) for which the above experiment satisfies the following requirements.*

- Correctness: *For any n and environment $\mathcal{E}$ generating $d, q \in F^n$ as above, the receiver's output at the end of the decommitment phase is $a = f_d(q) (= \langle d, q \rangle)$.*

- Binding: *For any environment $\mathcal{E}$ and efficient (but possibly non-uniform) malicious sender S^*, we define the following modified experiment. Run the commitment phase as before, except that S^* plays the role of S. Now, invoke the decommitment phase with R and S^** twice*, using identical inputs z_R, z_S, q but independently chosen random inputs. We say that S^** wins *if R outputs in the two invocations two* distinct *values a, a' such that $a, a' \in F$. We say that the protocol is* binding *if for every environment $\mathcal{E}$ and efficient S^*, the probability of S^* winning is negligible in n (where the probability is taken over the random inputs of the commitment phase and the two independent choices of random inputs of the decommitment phase).*

Note that, by a standard averaging argument, quantifying the above definition over all deterministic (non-uniform) environments is equivalent to quantifying over all probabilistic environments. The former is more convenient for proving the security of an implementation of this primitive, whereas the latter is more convenient when applying it in the context of higher level protocols.

Finally, it will be useful to rely on the following more intuitive notion of binding, which was already stated above: For every efficient cheating sender S^* there is a (possibly inefficient) *extractor* Ext which given the views of S^* and R in the commitment phase "extracts" a function $\tilde{f} : F^n \to F$ to which S^* is effectively committed. Formally:

Lemma 3.2 *Let (S, R) be a protocol for commitment with linear decommitment. Then, for every efficient S^* there is a function Ext such that the following holds. For any environment $\mathcal{E}$, the output of R at the end of the decommitment phase is guaranteed to be either $\tilde{f}(q) \stackrel{def}{=} \mathsf{Ext}(v_{S^*}, v_R, q)$ or $\perp$ except with negligible probability in n, where v_{S^*} and v_R are the views of S^* and R in the commitment phase, q is the decommitment query generated by $\mathcal{E}$, and the probability is over the random inputs of S^* and R in* both *phases.*

Proof: For any $a \in F$ and possible view $v = (v_{S^*}, v_R)$ in the commitment phase, let $A_v(q, a)$ denote the probability (over the randomness of the decommitment phase) that R outputs a. Given a view v we define $\tilde{f}(q) = \mathsf{Ext}(v, q)$ to be a field element a that maximizes $A_v(q, a)$. Assume towards a contradiction that Ext does not satisfy the required property. Then there is an environment $\mathcal{E}$ (producing inputs $d, q \in F^n$ for every n), a polynomial $p(\cdot)$, and infinitely many n, for which the following holds. With probability at least $1/p(n)$ over the randomness of the commitment phase (which determines the views v), we have $\sum_{a \in F \setminus \tilde{f}(q)} A_v(q, a) \geq 1/p(n)$ (where q is the query produced by $\mathcal{E}$). We show that one can partition F into two sets, each occurring with at least $1/3p(n)$ probability. Case 1: $A_v(q, \tilde{f}(q)) \geq 1/3p(n)$. In this case, the partition $(\tilde{f}(q), F \setminus \tilde{f}(q))$ will do. Case 2: $A_v(q, \tilde{f}(q)) < 1/3p(n)$. In this case all probabilities are smaller than $1/3p(n)$ but their sum is at least $1/p(n)$, which also implies the existence of a partition as required. It follows that S^* wins with non-negligible probability in the binding experiment of Definition 3.1. ■

Remark 3.3 (On hiding.) In contrast to standard cryptographic commitments, here we do not explicitly require a *hiding* property. The only reason that the sender avoids sending d itself is to save on communication complexity. However, it is easy to modify our implementation so that it additionally achieves (statistical) hiding of d.

Remark 3.4 (On generic implementations.) A generic method to get a communication-efficient implementation of our commitment primitive is the following. S commits to d by sending $c = H(d)$, where H is a collision-resistant hash function. In the decommitment phase, S sends $a = \langle d, q \rangle$ and uses (a PCP-based) efficient argument to prove that it knows d which satisfies $H(d) = c$ and $\langle d, q \rangle = a$. Our goal is to obtain a direct and more efficient implementation that avoids the use of short PCPs and makes a black-box use of the underlying cryptographic primitive. A natural approach would be to combine an efficient data structure with a tree-based commitment (that allows to efficiently open selected bits of the committed string) as follows. S commits to d by: (1) transforming d into an efficient data structure D that allows to answer any decommitment query q by probing few bits of D, and (2) committing to D using a tree-based commitment. A decommitment query q is then answered by opening just those (few) bits of D that are probed. Unfortunately, no efficient data structure can support arbitrary linear queries. This should be contrasted with the case of set membership queries and similar types of queries that have been previously addressed in related contexts [27, 29].

3.2 Efficient Implementation

In this section we give a direct efficient implementation for the basic variant of commitment with linear decommitment defined above. Our protocol is described in Figure 3.1.

BasicCommit + BasicDecommit

Commitment phase

Building block: a homomorphic encryption scheme $E = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ over a finite field F.

Sender's input: a vector $d \in F^n$ that defines a linear function $f_d : F^n \to F$ where $f_d(q) \stackrel{\text{def}}{=} \langle q, d \rangle$.

Receiver's input: length parameter n, computational security parameter k.

1. R generates the public and secret keys $(pk, sk) \leftarrow \mathsf{Gen}(1^k)$.
   It also generates a random vector $r \in_R F^n$ and encrypts (each entry of) r using Enc.
   It sends $\mathsf{Enc}(pk, r) = (\mathsf{Enc}(pk, r_1), \ldots, \mathsf{Enc}(pk, r_n))$ to S along with pk.

2. S uses the homomorphism of E to compute $e \in \mathsf{Enc}(pk, f_d(r))$ (without knowing r) and sends e to R.
   R decrypts the message e, letting $s \leftarrow \mathsf{Dec}(sk, e)$, and keeps s along with the vector r for decommitment.

Decommitment phase

Sender's input: $d \in F^n$, as above.

Receiver's inputs: decommitment query $q \in F^n$, decommitment information $r \in F^n$ and $s \in F$.

1. R picks a secret $\alpha \in_R F$ and sends the pair $(q, r + \alpha q)$ to S. (Each element of this pair is a vector in F^n.)

2. S responds with a pair $(a, b) = (f_d(q), f_d(r + \alpha q))$.
   R verifies that $b = s + \alpha a$; if so it outputs a; otherwise, it rejects (i.e., outputs $\perp$).

Figure 3.1: Basic commitment with linear decommitment

Correctness: If both parties behave honestly then the decrypted value s, obtained by the receiver in Step 2 of the commitment phase, satisfies $s = f_d(r)$. Furthermore, using the linearity of f_d, we have $b = f_d(r+\alpha q) = f_d(r) + \alpha \cdot f_d(q) = s + \alpha a$; thus, the verification succeeds and, moreover, the receiver outputs $a = f_d(q)$.

Communication complexity: The messages going from the receiver to the sender (in both phases) consist of $O(n)$ encrypted field elements, and the communication from the sender includes only $O(1)$ encrypted elements.

Before proving the binding property of our protocol, we present the following technical claim that is useful for our proof. In general, our proof shows that if binding does not hold then the encryption E is not secure. Rather than actually inverting E, it is enough to show that certain information about the plaintext can be computed, contradicting the *semantic security* of E.

Claim 3.5 *Fix any query sequence $\{q_n\}$, where $q_n \in F^n$.*

1. *Given* $(\mathsf{Enc}(r), r + \alpha q_n)$*, for random* $r \in F^n$ *and* $\alpha \in F$*, it is hard to compute* α*. Formally, for every efficient (nonuniform) algorithm* $\mathcal{A}$*, we have*

$$\Pr[\mathcal{A}(pk, \mathsf{Enc}(pk, r), r + \alpha q_n) = \alpha] \leq \frac{1}{|F|} + \mathrm{neg}(n),$$

where the probability is over $r \in_R F^n, \alpha \in_R F$ *and the randomness of* Gen *and* Enc*.*

2. *(Slightly more general.) For every efficient* $\mathcal{A}$*, we have* $\Pr[\mathcal{A}(pk, \mathsf{Enc}(pk, r), r + \alpha q_n, r + \alpha' q_n) = \alpha] \leq \frac{1}{|F|} + \mathrm{neg}(n)$*, where the probability is over* $r \in_R F^n, \alpha, \alpha' \in_R F$ *and the randomness of* Gen *and* Enc*.*

The proof of this claim follows directly from an appropriate formulation of semantic security (cf. [17], Def. 5.2.1).

Binding: We start by proving a weaker binding property, that holds with respect to a fixed random α picked by R in the decommitment phase. (In the binding experiment from Definition 3.1, this corresponds to fixing the random input of R in the two invocations of the decommitment phase.[2]) We argue that in the decommitment phase an efficient cheating S^* cannot come up, given $(q, r + \alpha q)$, with two pairs (a, b) and (a', b') such that both pass the verification by the receiver, with respect to r, s, q and α, but lead to different outputs (i.e., $a \neq a'$ and $a, a' \in F$). Whenever S^* produces such pairs, we say that S^* "wins". We want to show that S^* can only win with negligible probability.

The weak binding of the protocol is argued as follows. Assume that the above requirement does not hold; namely, there exists an efficient S^* for which there exists a query sequence q_n on which S^* wins the above game with non-negligible probability. We use q_n and S^* to construct a (non-uniform) algorithm $\mathcal{A}$ that will contradict Claim 3.5 (Part 1). Namely, $\mathcal{A}$ has the query $q = q_n$ hard-wired, it is given as input a public key pk, an encryption $\mathsf{Enc}(pk, r)$ and a vector $\beta = r + \alpha q$, and it guesses α with a non-negligible probability. Assuming $|F(n)|$ is super-polynomial in n, we have the required contradiction. Algorithm $\mathcal{A}$ works in four steps, corresponding to the four steps of the protocol, as follows: (a) $\mathcal{A}$ gives S^* the values pk and $\mathsf{Enc}(pk, r)$. (b) S^* gives back a value e which $\mathcal{A}$ ignores. (c) $\mathcal{A}$ gives S^* the pair (q, β). (d) $\mathcal{A}$ gets back two pairs (a, b) and (a', b'). Consider the event that S^* wins; this implies that $a \neq a'$, $b = s + \alpha a$ and $b' = s + \alpha a'$ (algorithm $\mathcal{A}$ need not know if this event occurs; it proceeds as if it does). Hence, $\mathcal{A}$ can compute $\alpha = (b - b')/(a - a')$ (note that $\mathcal{A}$ knows all the values in this expression).

The case of standard binding is similar; this time we will derive a contradiction to Part 2 of Claim 3.5. The modified algorithm $\mathcal{A}$ receives input $(pk, \mathsf{Enc}(pk, r), \beta, \beta')$, where $\beta = r + \alpha q$ and $\beta' = r + \alpha' q$; it starts the first two steps as before; then, $\mathcal{A}$ provides S^* (in the third step) a pair (q, β) and gets in return (in the last step) a pair (a, b); next, $\mathcal{A}$ repeats these two steps by sending another pair (q, β') and receiving back (a', b'). Now, algorithm $\mathcal{A}$ locally computes $\Delta = \alpha' - \alpha = (\beta'_j - \beta_j)/q_j$, where j is some index such that $q_j \neq 0$ (if $q = \vec{0}$ then $f_d(q) = 0$ and the protocol need not be invoked at all). Finally, $\mathcal{A}$ considers the equations $b = s + \alpha a$ and $b' = s + \alpha' a'$ (assuming, again, that it is in the event that S^* wins) and substitutes α' by $\alpha + \Delta$; i.e., $b' = s + (\alpha + \Delta)a'$. It now successfully computes $\alpha = (b' - b - \Delta \cdot a')/(a' - a)$.

[2] The difference between the weak and the standard versions of binding can be viewed as requiring the binding in two different places in the protocol: at the end of the Commitment phase (binding) or after the first step of the Decommitment phase (weak binding).

3.3 Multiple Commitments

In this section we extend the previous protocol to implement a more general commitment primitive, supporting multiple (parallel) commitments and linear decommitments. Specifically, the sender has ℓ (linear) functions, represented by vectors $d^1, \ldots, d^\ell \in F^n$ as before, and the receiver has ℓ queries $q^1, \ldots, q^\ell \in F^n$. For each query q^i, the receiver should get the answer $f_{d^i}(q^i) = \langle q^i, d^i \rangle \in F$. The protocol should satisfy the same property as the basic commitment protocol; namely, that every sequence of queries determines a unique answer. Moreover, we require that the answer to each query q^i depend on that query alone, and not on the other $\ell - 1$ queries. Intuitively, when going from MIPs to arguments this requirement will allow to mimic the "independence of provers" property of the MIP model.

A formal definition of the general binding property can be obtained by extending the experiment from Definition 3.1 as follows. In the commitment phase, $\mathcal{E}$ hands to the sender ℓ linear functions specified by $d^1, \ldots, d^\ell \in F^n$, and in the decommitment phase it hands to the receiver ℓ queries $q^1, \ldots, q^\ell \in F^n$. In the modified experiment used for defining the binding property, $\mathcal{E}$ hands to R two, possibly distinct, ℓ-tuples of queries $Q = (q^1, \ldots, q^\ell)$ and $\hat{Q} = (\hat{q}^1, \ldots, \hat{q}^\ell)$ in the two (independent) invocations of the decommitment phase. Denote by $A, \hat{A} \in F^\ell$ the two ℓ-tuples of answers. We say that S^* wins if, for some t, we have $q^t = \hat{q}^t$ but $a^t \neq \hat{a}^t$ (with $a^t, \hat{a}^t \in F$). The protocol is binding if for every $\mathcal{E}$ and efficient S^*, the probability that S^* wins is negligible in n.

In Figure 3.2 we describe in detail the generalized protocol. The protocol can be viewed as applying the protocol BasicCommit from the previous section (Figure 3.1) ℓ times in parallel, with an independent r for each d, q.

Correctness is argued as before. The communication complexity is again essentially optimal, and involves $O(\ell n)$ (encrypted) field elements sent from R to S and $O(\ell)$ elements sent in return. We turn to prove the generalized binding property.

Binding: Assume, for contradiction, that there exist an environment $\mathcal{E}$ and an efficient cheating sender S^* that violate the binding of the protocol, in the sense described above. We construct an algorithm $\mathcal{A}$ that breaks the semantic security of the encryption E (as in Claim 3.5). Namely, for infinitely many values of n (and hence also the security parameter k), the following holds. After the commitment phase is over, and the values $r^1, \ldots, r^\ell$ and $s^1, \ldots, s^\ell$ are already fixed, there exist $Q = (q^1, \ldots, q^\ell)$ and $\hat{Q} = (\hat{q}^1, \ldots, \hat{q}^\ell)$, with $q^t = \hat{q}^t = q$ (for some $t = t_n$), such that the cheating sender S^* has a non-negligible probability to pass the verification by the receiver with two tuples of ℓ pairs $((a^1, b^1), \ldots, (a^\ell, b^\ell))$ and $((\hat{a}^1, \hat{b}^1), \ldots, (\hat{a}^\ell, \hat{b}^\ell))$, respectively, that lead to different outputs on the t-th query (i.e., $a^t \neq \hat{a}^t$).

Algorithm $\mathcal{A}$ on input $(pk, \mathsf{Enc}(pk, r), \beta, \beta')$, where $\beta = r + \alpha q$ and $\beta' = r + \alpha' q$, guesses α with non-negligible probability as follows. (For each n, algorithm $\mathcal{A}$ has the tuples $Q, \hat{Q}$ and the index $t = t_n$ hard-wired as advice.) (a) $\mathcal{A}$ gives S^* the key pk along with a sequence of ℓ encryptions containing its input, $\mathsf{Enc}(pk, r)$, in the t-th position and encryptions of random $r^i \in_R F^n$ in the other $\ell - 1$ positions. (b) S^* gives back values $e^1, \ldots, e^\ell \in F$ which $\mathcal{A}$ ignores. (c) $\mathcal{A}$ gives S^* a sequence of ℓ pairs $(q^i, r^i + \alpha^i q^i)$, with (q, β) serving as the t-th pair and the other $\ell - 1$ pairs chosen as in the protocol, and gets back ℓ pairs (a^i, b^i). (d) $\mathcal{A}$ repeats the last step by giving S^* a sequence of ℓ pairs $(\hat{q}^i, r^i + \hat{\alpha}^i \hat{q}^i)$, where this time (q, β') serves as the t-th pair and the other $\ell - 1$ pairs are again chosen as in the protocol; $\mathcal{A}$ gets back ℓ pairs $(\hat{a}^i, \hat{b}^i)$. Consider the event that S^* wins; this implies that $a^t \neq \hat{a}^t$ and that all verifications succeeded; in particular, the verification related to the t-th pair implies that $b^t = s^t + \alpha a^t$ and $\hat{b}^t = s^t + \alpha' \hat{a}^t$ (again, $\mathcal{A}$ does not know if this event occurs; it proceeds as if it does). Now, $\mathcal{A}$ recovers α from $q, \beta, \beta', a^t, \hat{a}^t, b^t, \hat{b}^t$ as before.

Similarly the case of basic commitment, the original definition of binding implies the following more intuitive property: for every efficient cheating sender S^* there is a (possibly inefficient) extractor Ext which given the views of S^* and R in the commitment phase "extracts" an ℓ-tuple of functions $\tilde{f}^i : F^n \to F$ to which S^* is effectively committed. Formally, we have the following extension of Lemma 3.2:

Lemma 3.6 *Let (S, R) be a protocol for multiple commitments with linear decommitments. Then, for every efficient S^* there is a function* Ext *such that the following holds. For any environment $\mathcal{E}$, the output of R at the end of the decommitment phase is either $\perp$ or satisfies $a^i = \tilde{f}^i(q^i) \stackrel{def}{=} \mathsf{Ext}(i, v_{S^*}, v_R, q^i)$ for* all *i, except with $\mathrm{neg}(n)$ probability, where v_{S^*} and v_R are the views of S^* and R in the commitment phase, $(q^1, \ldots, q^\ell)$ are the decommitment queries generated by $\mathcal{E}$, and the probability is over the random inputs of S^* and R in* both *phases.*

Proof sketch: The proof is very similar to the proof of Lemma 3.2. Given q^i and views v we let $\mathsf{Ext}(i, v, q^i)$ pick the $\ell - 1$ remaining queries q_j arbitrarily and output a value from F that occurs most often as the i-th output of R. Assume towards contradiction that some ℓ-tuple of queries has q^i as its i-th entry but yields a different i-th output with non-negligible probability. Similarly to Lemma 3.2, this implies that we have two query ℓ-tuples that agree in their i-th entry but lead to a different i-th output with non-negligible probability, contradicting the extended binding requirement. ■

4 From Linear MIPs to Efficient Arguments

In this section we use the above commitment primitive to turn any linear MIP protocol (P, V) into an argument system (P', V') with related efficiency. Specifically, suppose that we are given an ℓ-prover linear MIP protocol for an NP language L. Recall that in such a protocol P specifies for the honest provers ℓ linear functions $\pi_1, \ldots, \pi_\ell$ (depending on the input x and an NP-witness w). The verifier V, on input x, sends to each P_i a query $q^i \in F^n$ and gets in return $\pi_i(q^i) \in F$. Based on the ℓ answers, V either rejects or accepts. If indeed $x \in L$ and w is a valid witness then, using the ℓ specified functions, V always accepts. If $x \notin L$ then any collection of ℓ functions $\tilde{\pi}_1, \ldots, \tilde{\pi}_\ell$ cannot make V accept with probability greater than ϵ.

We construct an argument system (P', V') for L as follows. On inputs x and w:

1. P' and V' run the sub-protocol $\mathsf{MultiCommit}$ (from Figure 3.2), where P' commits to the ℓ functions $\pi_1, \ldots, \pi_\ell$ obtained by the MIP protocol on inputs x, w. The verifier V' stores the decommitment information for later use.

2. V' runs (locally) the MIP verifier V on input x, to obtain ℓ MIP queries $q^1, \ldots, q^\ell \in F^n$.

3. P' and V' run the sub-protocol $\mathsf{MultiDecommit}$, using $q^1, \ldots, q^\ell$ as decommitment queries (V' also uses the decommitment information stored above). V', playing the receiver in this sub-protocol, either rejects or it gets the values $\pi_1(q^1), \ldots, \pi_\ell(q^\ell)$ to which it again applies the MIP verifier V and accepts/rejects accordingly.

Theorem 4.1 *Suppose (P, V) is an $\epsilon(n)$-sound linear MIP protocol over a field $F(n)$ such that $|F(n)| = n^{\omega(1)}$. Then (P', V') described above is an $\epsilon'(n)$-sound argument protocol for L, where $\epsilon'(n) \leq \epsilon(n) + \mathrm{neg}(n)$.*

Proof: The completeness of (P', V') follows directly from the completeness of the underlying MIP (P, V) and the correctness of the commitment protocol. Soundness follows from the soundness of (P, V) and the binding of the commitment protocol (where the latter requires $|F(n)| = n^{\omega(1)}$). Specifically, suppose $x \notin L$. The extractor Ext guaranteed by Lemma 3.6 defines, at the end of Step 1, effective proof functions $\tilde{\pi}^1, \ldots, \tilde{\pi}_\ell$ such that, except with $\mathrm{neg}(n)$ probability, the answers obtained by V' in Step 3 are $(\tilde{\pi}^1(q^1), \ldots, \tilde{\pi}_\ell(q^\ell))$ (unless V' rejects during decommitment). By the ϵ-soundness of (P, V), the probability that V' accepts is at most $\epsilon(n) + \mathrm{neg}(n)$ as required. ■

Complexity: If each of the ℓ provers of the MIP protocol computes a function $\pi_i : F^n \to F$ then the complexity of the argument system consists of $O(\ell \cdot n)$ encrypted field elements from the verifier to the prover, and $O(\ell)$ encrypted field elements from the prover to the verifier.

Moving communication to an offline phase. We now describe a heuristic approach for moving almost all of the communication sent by the verifier to an offline preprocessing phase, before the input x is known. This results in a very efficient online phase. The protocol generated by the compiler has the following high level structure: it starts with the verifier sending a message ("query") q_1. The prover responds with an answer a_1 that depends on x. The verifier sends another query q_2 and the prover responds with another answer a_2 that depends on x. Note that neither q_1 nor q_2 depend on x. Moreover, typically both a_1 and a_2 will be very short, whereas q_1, q_2 will be long. The idea is to have the verifier send in an offline phase the original q_1 as well as an "encryption" of q_2. In the online phase, after the input is revealed, the prover responds with a_1. Then the verifier reveals to the prover the (short) key it used to encrypt q_2, and the verifier responds with a_2. The soundness of this approach can be proved in the random oracle model. Intuitively, in this model it is easy to guarantee that the answer a_1 is statistically independent of the encrypted query q_2. We leave open the question of instantiating the encryption function using standard cryptographic assumptions.

5 Constructing Linear MIPs

In this section we give a self-contained presentation of a simple linear MIP based on Hadamard codes that can be used to instantiate the argument protocol from the previous section. The MIP protocol is obtained by applying a standard transformation to the Hadamard based linear PCP protocol from [1], Section 6 (see also [6], Appendix A).

Towards a modular presentation, it will be convenient to use the following intermediate proof models. A *linear PCP* is defined similarly to linear MIP except that honest provers all use the same (linear) function π, and soundness should only hold with respect to a single $\tilde{\pi}$ (such that $\tilde{\pi}_i = \tilde{\pi}$ for all $1 \leq i \leq \ell$). We also define a variant of linear PCP called *weak linear PCP* in which soundness is only guaranteed to hold when false proofs $\tilde{\pi}$ are linear.

Below we show, in a sequence of simple steps, how to construct linear MIPs. Specifically, we start with a weak linear PCP as above. Then, by adding a simple linearity test (but avoiding the "low-degree test" commonly used in PCP constructions), we get a linear PCP protocol. Finally, we turn the linear PCP into a linear MIP by splitting the role of the proof among several provers and adding a standard consistency test. In all steps, we use the following notations. Let $x \in \{0,1\}^u$ be an input to an NP-statement (of the form $x \in L$) and w be a corresponding witness. Let $C = C(x, w)$ be a circuit of size s (gates) testing the validity of w with respect to x. With each gate j of C, we associate a variable Z_j. The satisfiability of C by x, w can now be expressed as a conjunction of the following conditions: (1) u conditions, corresponding to the u input gates labelled by x, and having the form $Z_i = x_i$, for $i \in [u]$ (testing the consistency of the

first u input gates with the actual input x; there are no similar constraints on input gates labelled by w); (2) conditions corresponding to internal gates of the circuit, that have the form $(1 - Z_i Z_j) - Z_k = 0$ (testing the consistency of Z_k, the output of a NAND gate, with its two inputs coming from gates Z_i, Z_j); and finally (3) a condition of the form $Z_s = 1$ (testing that the output of the circuit is 1). To get rid of the last condition, we can simply fix Z_s to 1. Altogether, there are $m < s$ constraints.

A weak linear PCP (with $n = O(s^2)$). Let z be the assignment for $Z_1, \ldots, Z_s$ (i.e., the actual values of all gates in the circuit on input (x, w), as above). The proof oracle π can be written as a linear function f_d, where $d = (z, z \otimes z)$, and where $a \otimes b$ denotes the concatenation of all $|a| \cdot |b|$ values $a_i b_j$. (In the standard PCP terminology, the proof is a Hadamard encoding of d.) Verifying the proof involves verifying that d is indeed of the form $(z, z \otimes z)$ (for some z) and, moreover, that z satisfies all the above conditions.

To verify the form of d, the verifier picks at random $y_1, y_2 \in_R F^s$ and verifies that $\langle z, y_1 \rangle \cdot \langle z, y_2 \rangle = \langle z \otimes z, y_1 \otimes y_2 \rangle$ (this boils down to evaluating 3 linear combinations of the entries of d, which can be done using 3 queries to π). If π is of the claimed form then it always passes the test, while if π is not of this form then the test fails (and hence the proof is rejected) with probability at least $1/4$. [3]

Next, the verifier tests that z passes the conditions regarding the circuit (here it is convenient to assume that the proof π already passed the first test and that it is of the required form). Note that each of the m conditions can be expressed in the form $Q_i(z) = 0$, where Q_i is a multivariate polynomial (in $Z_1, \ldots, Z_s$) of degree at most 2. A linear test whether $Q_1, \ldots, Q_m$ all equal 0 works by picking $v \in_R F^m$ and verifying that $Q_v(z) \stackrel{\text{def}}{=} \sum_{i=1}^m v_i \cdot Q_i(z) = 0$. If each $Q_i(z)$ is indeed 0 then so is $Q_v(z)$, while if for at least one i we have $Q_i(z) \neq 0$ then the probability that $Q_v(z) = 0$ is $1/|F|$. Note that Q_v itself is a degree-2 polynomial and hence asking for the value of $Q_v(z)$ can also be represented as making a linear query to f_d, where $d = (z, z \otimes z)$.

To conclude, each query in the above PCP proof is of the form $q \in F^n$, for $n = s^2 + s$, and the answer of an honest prover is of the form $\langle d, q \rangle$. The (weak) soundness relies on the fact that a false proof $\tilde{\pi} = f_{\tilde{d}}$ must be caught by either of the two tests described above, except with (at most) constant probability.

From *weak* linear PCP to linear PCP. Given a verifier V of a weak linear PCP, where the soundness is guaranteed only under the assumption that false "proofs" are still linear, we turn it into another verifier V' with soundness that does not make any such assumption. This transformation uses the standard testing and self-correcting approach.

As before, denote the proof by $\pi : F^n \to F$. As a first step, we turn the weak PCP into a *smooth* (but still weak) one; that is, where each query q is (separately) uniformly distributed in F^n. To turn a non-smooth weak PCP into a smooth weak PCP, the verifier V replaces each query q by a pair of queries q_1, q_2 which are random subject to $q_1 + q_2 = q$. Given the answers to these queries, V computes $\pi(q) = \pi(q_1) + \pi(q_2)$ (note that both "correct" proofs π and "false" proofs $\tilde{\pi}$ are linear in this case so this transformation does not affect acceptance probabilities). Hence, from now on, we may assume that the given weak PCP is already smooth. Loosely speaking, V' works as V except that it starts with a "linearity test" of the proof π. That is, V' picks at random $q_1, q_2 \in F^n$, asks for $\pi(q_1), \pi(q_2)$ and $\pi(q_1 + q_2)$ and verifies that $\pi(q_1) + \pi(q_2) = \pi(q_1 + q_2)$. If π is linear then it always passes the test, while if V' is given a proof $\tilde{\pi}$ which is δ-far from linear, then

[3] The argument goes as follows: given $\tilde{d} = (z, U)$, think of U as an $s \times s$ matrix and let $V = z \otimes z$ be another $s \times s$ matrix. The verifier can be viewed as testing whether $U = V$. The test essentially compares $y_1 U y_2$ to $y_1 V y_2$. If $U \neq V$ (which may occur in a single position) then, with probability at least $1/2$, the vectors $y_1 U, y_1 V$ are different and therefore $(y_1 U) y_2 \neq (y_1 V) y_2$ with probability at least $1/4$. Note that we gain more if F is large as, in fact, the success probability is $(1 - 1/|F|)^2$. Alternatively, repeating the test amplifies the probability as needed.

this is caught with probability δ [10]. Therefore, if the proof is δ-far from linear, then V' is likely to "catch" this, while if it is δ-close to some linear $\tilde{\pi}'$ then V' is likely not to ask any question where $\tilde{\pi}$ and $\tilde{\pi}'$ disagree. Specifically, since we assumed V to be smooth, and denoting by ℓ the number of queries it makes, then except with probability $\delta\ell$ the new V' only asks queries q where the answer $\tilde{\pi}(q)$ is identical to the answers of the linear $\tilde{\pi}'(q)$. Hence, by the soundness of V with respect to a linear $\tilde{\pi}'$, the desired soundness follows.

From linear PCP to linear MIP. Suppose that we are given a linear PCP protocol with a verifier V' that asks ℓ' queries. We construct a linear MIP protocol for a verifier V'' and $\ell'+1$ provers as follows. Verifier V'' works as V' does but it sends each of its ℓ' queries to a different prover. In addition, it picks one of the ℓ' queries at random and asks Prover $(\ell'+1)$ about it. The verifier V'' accepts if the answer from the last prover is consistent with the previous answers and, in addition, V' would accept with these answers. As before, completeness is trivial. For the soundness, we can assume without loss of generality that the provers are deterministic. Hence, the answers provided by prover $P_{\ell'+1}$ on each query q define a proof $\tilde{\pi}$ (where $\tilde{\pi}(q)$ is its answer to query q). The idea is that, whenever the answers of the first q' provers are consistent with this $\tilde{\pi}$ then their cheating probability is bounded by the soundness of the linear PCP with proof $\tilde{\pi}$. On the other hand, whenever at least one of the answers is inconsistent with $\tilde{\pi}$, this will be caught with probability at least $1/\ell'$.

To improve the soundness probability to $2^{-\sigma}$, we can use additional provers, either by simple independent repetitions of the above MIP (which will require $O(\sigma)$ provers) or via more efficient techniques. See [32] for more on transforming PCP to MIP.

6 Beyond Laconic Provers

In this section we sketch extensions of the basic construction from Section 4 that improve its total communication complexity without relying on preprocessing. We note that we view the basic construction above as where the conceptual contribution of this paper is and, moreover, similar ideas to those used here were already used in various forms in the literature (e.g., [12, 24]).

The two techniques we use for improving the communication complexity are termed *balancing* and *recursion*. The main idea behind *balancing* is the observation that in the basic construction the verifier sends long messages (vectors of $n = O(s^2)$ field elements) and gets in return a single field element per query. Balancing aims at reducing the communication from the verifier to the prover at the expense of increasing the communication in the other direction so as to "balance" between the two. This step is done at the PCP/MIP level.

Recursion is done at the arguments level. The idea behind (one level of) recursion is to let the prover send "compressed" answers instead of its original answers, and then (after the normal execution is complete) let it recursively prove using an efficient argument that it knows the (long) answers that are consistent with the compressed answers and would make the verifier accept. An appropriate combination of balancing and recursion leads to communication complexity of $O(n^\epsilon)$ using $O(1/\epsilon)$ rounds, for an arbitrary constant $\epsilon > 0$.

6.1 Balancing

We now provide further details on the balancing technique. In the argument protocol obtained by applying the general compiler from Section 4 to the linear MIP from Section 5 the verifier sends "long" queries of $n = O(s^2)$ field elements and gets $O(1)$ field elements in return. Our balancing starts at the level of *weak*

linear PCPs, where we first construct so-called "template-based" weak linear PCPs and then we show how to transform them to linear MIPs while maintaining their complexity.

In a *template-based* weak PCP, we cover the n entries of the vector $d \in F^n$ representing the proof π by (overlapping) sets $S_1, \ldots, S_\beta$ each of size $|S_i| = \delta \ll n$; on a query $q \in F^\delta$ the (linear) answer is $\langle d^1, q\rangle, \ldots, \langle d^\beta, q\rangle$, where each d^i (referred to as a "block") is the sub-vector of d whose coordinates are in S_i. (Note that each element of the answer, $\langle d^i, q\rangle$, can still be viewed as a linear function of d itself, i.e. $\langle d, q'\rangle$, where the query $q' \in F^n$ is restricted to have non-zero elements only in the subset S_i; hence we are still in the same framework and, in particular, in the "weak" version the cheating proof still consists of one linear function $\tilde{\pi}$.) We emphasize that the partition $S_1, \ldots, S_\beta$ is *fixed* and, in particular, is independent of the query q. Moreover, we may have a different partition for each of the queries in the protocol (as long, as the partition is fixed and independent of q).

Recall that our basic weak linear PCP consists of a proof $\pi = f_d$ where $d = (z, z \otimes z)$, and the verifier tests essentially that (1) the first u bits in z are equal to x; (2) π is indeed of the form $(z, z \otimes z)$; and (3) for all i, $Q_i(z) = 0$. We show how to turn each of the three tests into template-based tests. Dealing with the first test is simple: rather than ask for the value $\langle z_1 \ldots z_u, q\rangle$, for $q \in_R F^u$, we view $z_1 \ldots z_u$ (and x) as consisting of β blocks $z^1, \ldots, z^\beta$ of size u/β each (in this case, the blocks are actually disjoint). On a query $q \in_R F^{u/\beta}$, the answer is a vector of β field elements: $\langle z^1, q\rangle, \ldots, \langle z^\beta, q\rangle$. The verifier compares this vector with $\langle x^1, q\rangle, \ldots, \langle x^\beta, q\rangle$ and rejects if they are not equal. Clearly, if $z_1 \ldots z_u \neq x$ then, for at least one block we have $z^i \neq x^i$, and this is caught with probability $1 - 1/|F|$.

Next, we deal with the second test. Recall that $\pi = (z, U)$, where U is an $s \times s$ matrix and that the test compares U to $V \stackrel{\text{def}}{=} z \otimes z$. A first attempt is to proceed as in the previous test; specifically, rather than pick $q_1, q_2 \in_R F^s$, partition the s rows of the matrix to λ sets of size s/λ and similarly for the columns (as well as for z itself). Then, pick $q_1, q_2 \in_R F^{s/\lambda}$ and verify, for each of the sub-matrices, that $\langle z^i, q_1\rangle \cdot \langle z^j, q_2\rangle = \langle U^{i,j}, q_1 \otimes q_2\rangle$, where $U^{i,j}$ denotes the (i,j) sub-matrix of U and z^i, z^j denote, as before, the i-th and j-th blocks (respectively) of z. While this modification works, it does not give us anything in terms of complexity as we have λ^2 sub-matrices (this corresponds to the length of the answer β) each containing s^2/λ^2 elements (which correspond to δ), and the product of these two parameters is s^2. To overcome this problem, we will assume that C is a *repetitive* circuit; namely, we assume that C can be partitioned into β (possibly overlapping) sub-circuits $C_1, \ldots, C_\beta$ (each of size $\delta \approx s/\beta$ gates) where each internal gate appears together with its two inputs in one of the sub-circuits; in fact (for the sake of third test only), we will further assume that each of the β sub-circuits has the same topology.[4] Denote by $A_i \subseteq [s]$ the set of gates that participate in sub-circuit C_i (and $[s] = \cup_{i\in[\beta]} A_i$). It suffices to test that each U_i, which is the $A_i \times A_i$ sub-matrix of U (and where S_i consists of the corresponding $|A_i|^2$ coordinates in d), is consistent with z. [5]

Finally, we deal with the third test. Again, we use the partition of C to sub-circuits $C_1, \ldots, C_\beta$ and, moreover, we take advantage of the assumption, mentioned above, that each of the sub-circuits has the same topology[6] to apply the same test to each of them. Namely, rather than picking a random combination of the

[4]Such a circuit can be obtained in various ways: from a TM via Ladner's theorem; by applying an appropriate transformation to an arbitrary circuit such as using selectors or sorting networks; or, most importantly, by applying recursion to previous circuits (as we will do later) in a way that maintains such property.

[5]Note that these sub-matrices overlap; however, since they are all tested against the same z they are also consistent with each other. On the other hand, other entries of U (in $A_i \times A_j$, for $i \neq j$) will not be in use and, in fact, can be omitted from π.

[6]Specifically, we assume that for each pair of sub-circuits C_i, C_j the isomorphism is such that the k-th gate in S_i corresponds to the k-th gate in S_j.

polynomials corresponding to the s gates in C, we pick a random combination of δ polynomials and apply it (in a template-based fashion) to each of the β sub-circuits.

From *weak* linear PCP to linear PCP. The reason that the transformation described in Section 5 does not work here is that both turning the weak linear PCP into a smooth one and the linearity test actually ask queries on "heavy" vectors q that do not obey the template-based structure. To fix this, we modify the protocol in the natural way; namely, the smoothening is done within the sets of the templates. That is, each query $q \in F^\delta$ is replaced by a pair of random $q_1, q_2 \in F^\delta$ with $q_1 + q_2 = q$ and then q_1, q_2 are answered in a template-based manner; this allows constructing the β answers to query q from the β answers to each of q_1, q_2. Similarly, we deal with the linearity testing (i.e., on $q_1, q_2 \in_R F^\delta$, for each of the β blocks the verifier compares the answers to query $q_1 + q_2$ to the sum of answers to queries q_1, q_2). This, however, is not good enough: it only guarantees that the "restriction" of $\tilde{\pi}$ on each S_i is close to some linear function f_i. This falls short of assuring that there is one linear function $\hat{\pi}$ (on all n variables) such that, for all i, the function $\hat{\pi}$ is close to $\tilde{\pi}$ when both are restricted to S_i. (Note that this does not imply that $\tilde{\pi}$ is close to $\hat{\pi}$.) To deal with this, we add another test. The idea is to verify, for each pair i, j such that $S_i \cap S_j \neq \emptyset$ that the corresponding f_i, f_j when restricted to $S_i \cap S_j$ are close. This is done by picking a random element in this intersection and testing it. To get the desired complexity, we observe that, for the same reasons that we can guarantee that C is repetitive, we can also guarantee that the number of intersecting pairs is only $O(\beta)$ (and in particular much smaller than all $\binom{\beta}{2}$ possibilities). In addition, since the S_i's are fixed then so are the their intersections and so we can use template queries to perform all these tests. The claim is that if we pass all these tests then $\hat{\pi}$ as above exists and hence we can continue running the verifier as in the *weak* PCP setting.

From linear PCP to linear MIP. To obtain a linear MIP protocol, we again split the ℓ' PCP queries among $\ell' \cdot \sigma$ provers (note that a template query with β answers is viewed here as β questions). The MIP verifier V'' runs the PCP verifier V' to obtain some ℓ' queries $q_1, \ldots q_{\ell'}$. It then, in a random order, asks each of the ℓ' queries to σ provers (each prover gets a single question). Then, V'' accepts if the answers are consistent with each other and the PCP verifier V' accepts with these answers. The completeness is trivial. To argue the soundness, we show how cheating provers in the MIP setting can be turned into a "cheating" PCP proof. Note that the provers are deterministic, so the answer of each prover P_i on each query q is well defined. We therefore define $\tilde{\pi}(q)$ to be the majority of the answers $P_1(q), \ldots, P_{\ell' \cdot \sigma}(q)$. Whenever all provers answer according to $\tilde{\pi}$ then the error probability is bounded by the PCP soundness (with respect to this $\tilde{\pi}$). Whenever the provers are inconsistent with $\tilde{\pi}$ on at least one query q_i but are still consistent among themselves, this means that the query q_i was assigned σ times to provers which are in the minority. This happens with $\mathrm{neg}(\sigma)$ probability.

From linear MIP to arguments. We describe a template-based version of the "MIP to arguments" transformation (from Section 4) that takes the template structure into account and results in the desired communication complexity. Specifically, we are given a template-based linear MIP protocol with ℓ linear functions (provers) $\pi_1, \ldots, \pi_\ell$, a template that consists of β sets $S_1, \ldots, S_\beta$ of size δ and a verifier V. We construct an argument system for prover P' and verifier V'; it starts by P' and V' running the sub-protocol MultiCommit, where P' commits to the $\ell \cdot \beta$ functions $\pi_{i,j}$ (for $i \in [\ell], j \in [\beta]$) where $\pi_1, \ldots, \pi_\ell$ are the ℓ (linear) functions specified by the linear MIP protocol and $\pi_{i,j}$ is the (linear) function induced by projecting π_i on the coordinates in S_j. Then, V' runs the MIP verifier, V, to obtain ℓ queries $q^1, \ldots, q^\ell \in F^\delta$ (each of which should be answered with respect to the β sets). Finally, P' and V' run the sub-protocol MultiDecommit, with the $\ell \cdot \beta$ functions $\pi_{i,j}$ and query q^i for each $\pi_{i,j}$ The verifier V', playing the receiver in this sub-protocol, either rejects or it gets the $\ell \cdot \beta$ values $\pi_{i,j}(q^i)$ to which it again applies the MIP verifier V and accepts/rejects

accordingly.

6.2 Recursion

The above balancing technique, applied to the Hadamard based PCP from Section 5, can give the following efficiency tradeoff: if the verifier's queries are of length $s^{2\gamma}$ then the answers are of length $\approx s^{1-\gamma}$. In particular, setting $\gamma = 1/3$, the total communication becomes roughly $s^{2/3}$. The idea for going below this barrier is to apply a recursive composition of argument protocols along the following lines. (For technical reasons, we need the arguments to satisfy a stronger *proof of knowledge* property that is indeed satisfied by our constructions.) We run the above argument protocol with parameter γ, except that the prover compresses its original answers (whose length is $\approx s^{1-\gamma}$) using a succinct commitment. After the execution is complete, the verifier sends its secret decryption key, given which the verifier's acceptance predicate becomes computable by a circuit of a nearly linear size in the prover's view (namely, of size $s' \approx s^{1-\gamma}$). Now the prover needs to convince the verifier that it "knows" the original long answers that are consistent with the hashed answers and would make the verifier accept. This is done (recursively) via an efficient argument that uses a parameter γ' with a verification circuit of size s'. Applying this composition $O(1/\epsilon)$ times with appropriate choices of γ yields an argument protocol with a total communication of $O(s^{\epsilon})$.

Acknowledgements. We are grateful to Eli Ben-Sasson and Prahladh Harsha for enlightening discussions on the efficiency of short PCPs, and to Rafael Pass for pointing out the relevance of [33]. We also thank the anonymous CCC referees for helpful suggestions and comments.

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof Verification and the Hardness of Approximation Problems. *J. ACM* 45(3): 501-555, 1998. Preliminary version in FOCS '92.

[2] S. Arora, and S. Safra. Probabilistic Checking of Proofs: A New Characterization of NP. *J. ACM* 45(1): 70-122, 1998. Preliminary version in FOCS '92.

[3] L. Babai, L. Fortnow, L. A. Levin, and M. Szegedy. Checking Computations in Polylogarithmic Time. In *STOC 1991*, pages 21-31.

[4] B. Barak. How to Go Beyond the Black-Box Simulation Barrier. In *FOCS 2001*, pages 106–115.

[5] B. Barak and O. Goldreich. Universal Arguments and their Applications. In *CCC 2002*, pages 194-203.

[6] E. Ben-Sasson, O. Goldreich, P. Harsha, M. Sudan, and S. P. Vadhan. Robust PCPs of proximity, shorter PCPs and applications to coding. *SIAM J. Comput.* 36(4): 889-974, 2006. Earlier version in STOC 2004.

[7] E. Ben-Sasson, O. Goldreich, P. Harsha, M. Sudan, and S. P. Vadhan. Short PCPs Verifiable in Polylogarithmic Time. In *CCC 2005*, pages 120-134.

[8] E. Ben-Sasson and M. Sudan. Simple PCPs with poly-log rate and query complexity. In *STOC 2005*, pages 266-275.

[9] G. Brassard, D. Chaum, and C. Crépeau. Minimum Disclosure Proofs of Knowledge. *J. Comput. Syst. Sci.* 37(2): 156-189, 1988. Preliminary version in FOCS '86.

[10] M. Blum, M. Luby, and R. Rubinfeld. Self-Testing/Correcting with Applications to Numerical Problems. *J. Comput. Syst. Sci.* 47(3): 549-595, 1993. Preliminary version in STOC '90.

[11] D. Boneh, E.J. Goh, and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. In *Proc. 2nd TCC*, pages 325–341, 2005.

[12] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. *J. of the ACM*, 45:965–981, 1998.

[13] I. Dinur. The PCP theorem by gap amplification. In *STOC 2006*, pages 241-250.

[14] I. Dinur and O. Reingold. Assignment Testers: Towards a Combinatorial Proof of the PCP-Theorem. In *FOCS 2004*, pages 155-164.

[15] T. ElGamal. A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, v. IT-31, n. 4, 1985, pages 469-472.

[16] U. Feige, S. Goldwasser, L. Lovasz, S. Safra, and M. Szegedy. Interactive Proofs and the Hardness of Approximating Cliques. *J. ACM* 43(2): 268-292, 1996. Preliminary version in FOCS '91.

[17] O. Goldreich. *Foundations of Cryptography: Basic Applications*. Cambridge University Press, 2004.

[18] O. Goldreich and J. Håstad. On the Complexity of Interactive Proofs with Bounded Communication. *Inf. Process. Lett.* 67(4): 205-214, 1998.

[19] O. Goldreich, S. P. Vadhan, and A. Wigderson. On interactive proofs with a laconic prover. *Computational Complexity* 11(1-2): 1-53, 2002.

[20] S. Goldwasser and S. Micali. Probabilistic encryption. *JCSS*, 28(2):270–299, 1984. Preliminary version in STOC '82.

[21] S. Goldwasser, S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM J. Comput.*, Vol. 18, No. 1, pp. 186-208, 1989.

[22] Y. Ishai, E. Kushilevitz and R. Ostrovsky. Sufficient Conditions for Collision-Resistant Hashing. In *Proc. 2nd TCC*, pages 445–456, 2005.

[23] J. Kilian. A Note on Efficient Zero-Knowledge Proofs and Arguments. In *STOC 1992*, pages 723-732.

[24] J. Kilian. Improved Efficient Arguments. In *CRYPTO 1995*, pages 311–324.

[25] R. C. Merkle. A Certified Digital Signature. In *CRYPTO 1989*, pages 218-238.

[26] S. Micali. Computationally Sound Proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2000.

[27] S. Micali, M. Rabin and J. Kilian. Zero knowledge sets. In *Proc. FOCS 2003*, pages 80-91.

[28] M. Naor and K. Nissim. Communication preserving protocols for secure function evaluation. In *STOC 2001*, pages 590–599.

[29] R. Ostrovsky, C. Rackoff, and A. Smith. Efficient Consistency Proofs for Generalized Queries on a Committed Database. In *Proc. ICALP 2004*, pages 1041-1053.

[30] P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *EUROCRYPT 1999*, pages 223-238.

[31] A. Polishchuk, and D. A. Spielman. Nearly-linear size holographic proofs. In *STOC 1994*, pages 194-203.

[32] A. Ta-Shma. A Note on PCP vs. MIP. *Inf. Process. Lett.* 58(3): 135-140, 1996.

[33] M. Zimand. Probabilistically Checkable Proofs the Easy Way. In *IFIP TCS 2002*, pages 337-351.

MultiCommit + MultiDecommit

Commitment phase

Building block: a homomorphic encryption scheme $E = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ over a finite field F.

Sender's input: ℓ vectors $d^1, \ldots, d^\ell \in F^n$ that define ℓ linear functions $f_{d^i} : F^n \to F$ where $f_{d^i}(q) \stackrel{\text{def}}{=} \langle q, d^i \rangle$.

Receiver's input: length parameter n, number of commitments ℓ, and a security parameter k.

1. R generates the keys $(pk, sk) \leftarrow \mathsf{Gen}(1^k)$.
   It also generates ℓ random vectors $r^1, \ldots, r^\ell \in_R F^n$ and encrypts $r^1, \ldots, r^\ell$ using Enc.
   It sends $\mathsf{Enc}(pk, r^1), \ldots, \mathsf{Enc}(pk, r^\ell)$ to the sender along with pk.

2. S uses the homomorphism of E to compute $e^1 \in \mathsf{Enc}(pk, f_{d^1}(r^1)), \ldots, e^\ell \in \mathsf{Enc}(pk, f_{d^\ell}(r^\ell))$ (using the d^i's and without knowing the r^i's) and sends $e^1, \ldots, e^\ell$ to R.
   For each i the receiver R lets $s^i \leftarrow \mathsf{Dec}(sk, e^i)$, and keeps s^i along with the vector r^i for the decommitment.

Decommitment phase

Sender's input: ℓ vectors $d^1, \ldots, d^\ell \in F^n$, as above.

Receiver's inputs: ℓ queries $q^1, \ldots, q^\ell \in F^n$, decommitment information $r^1, \ldots, r^\ell \in F^n$ and $s^1, \ldots, s^\ell \in F$.

1. R picks at random ℓ secrets $\alpha^1, \ldots, \alpha^\ell \in_R F$ and sends the ℓ vector pairs $(q^i, r^i + \alpha^i q^i)$ to S.

2. S responds with ℓ pairs $(a^i, b^i) = (f_{d^i}(q^i), f_{d^i}(r^i + \alpha^i q^i))$, where $a^i, b^i \in F$.
   R verifies, for each i, that $b^i = s^i + \alpha^i a^i$; if so it outputs $(a^1, \ldots, a^\ell)$; otherwise, it rejects (i.e., outputs $\perp$).

Figure 3.2: Parallel commitments with linear decommitments