Advisor-Verifier-Prover Games and the Hardness of Information Theoretic Cryptography*

Benny Applebaum  
Tel-Aviv University  
Tel-Aviv, Israel  
benny.applebaum@gmail.com

Oded Nir  
Tel-Aviv University  
Tel-Aviv, Israel  
odednir123@gmail.com

September 14, 2023

Abstract

A major open problem in information-theoretic cryptography is to obtain a super-polynomial lower bound for the communication complexity of basic cryptographic tasks. This question is wide open even for very powerful non-interactive primitives such as private information retrieval (or locally-decodable codes), general secret sharing schemes, conditional disclosure of secrets, and fully-decomposable randomized encoding (or garbling schemes). In fact, for all these primitives we do not even have super-linear lower bounds. Furthermore, it is unknown how to relate these questions to each other or to other complexity-theoretic questions.

In this note, we relate all these questions to the classical topic of query/space trade-offs, lifted to the setting of interactive proof systems. Specifically, we consider the following Advisor-Verifier-Prover (AVP) game: First, a function $f$ is given to the advisor who computes an advice $a$. Next, an input $x$ is given to the verifier and to the prover who claims that $f(x) = 1$. The verifier should check this claim via a single round of interaction based on the private advice $a$ and without having any additional information on $f$. We focus on the case where the prover is laconic and communicates only a constant number of bits, and, mostly restrict the attention to the simplest, purely information-theoretic setting, where all parties are allowed to be computationally unbounded. The goal is to minimize the total communication complexity which is dominated by the length of the advice plus the length of the verifier’s query.

As our main result, we show that a super-polynomial lower bound for AVPs implies a super-polynomial lower bound for a wide range of information-theoretic cryptographic tasks. In particular, we present a communication-efficient transformation from any of the above primitives into an AVP protocol. Interestingly, each primitive induces some additional property over the resulting protocol. Thus AVP games form a new common yardstick that highlights the differences between all the above primitives.

Equipped with this view, we revisit the existing (somewhat weak) lower bounds for the above primitives, and show that many of these lower bounds can be unified by proving a single counting-based lower bound on the communication of AVPs, whereas some techniques are inherently limited to specific domains. The latter is shown by proving the first polynomial separations between the complexity of secret-sharing schemes and conditional disclosure of secrets and between the complexity of randomized encodings and conditional disclosure of secrets.

---

*This is the full version of a paper that appears in FOCS’23. Research supported by ISF grant no. 2805/21 and by the European Union (ERC, NFITSC, 101097959). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.
1 Introduction

Information-Theoretic Cryptography deals with the task of secure computation and communication in the presence of computationally unbounded adversaries. On the foundational level, the goal is to understand the “cost” of realizing a cryptographic task where the main efficiency measures are communication and randomness. As an archetypal example, Shannon’s seminal work from the late 40s \[59\] defines the task of perfectly-secret communication and provides an upper-bound and a matching lower-bound on its cost. Unfortunately, for many other tasks, our understanding is much more limited and there are huge, typically exponential gaps, between the best-known upper bounds and lower bounds. Furthermore, this is true even for very simple primitives that involve a minimal amount of interaction. Let us review some of the notable examples.

- **Private Information Retrieval (PIR)** [26]: In a PIR scheme, a client wishes to query the \(x\)th location of some function \(f : \{0, 1\}^n \rightarrow \{0, 1\}\) that is being held by \(k\) computationally-unbounded non-colluding servers without revealing the location \(x\) to any single server. In the most popular setting the communication is restricted to a single round in which the client generates a vector of queries, one to each server, and receives back from the servers a vector of responses. We will further focus on the case of short-downstream PIR (sd-PIR) where the total communication complexity from the servers to the client is constant (i.e., the number of servers is constant and each server sends a constant number of bits) and the goal is to minimize the maximal message length sent by the client. This famous open problem is essentially equivalent to the question of determining the best-achievable rate of a smooth locally-decodable code (LDCs) with a constant query complexity and constant-size alphabet \[48, 38\]. (See the survey \[67\].) Currently, the best-known upper bounds for sd-PIR are of the form \(O(\sqrt{n})\) \[66, 33, 31\]. A lower bound of \(Cn\) for some constant \(C > 1\) is known since the late 90s \[54, 63\] and while the constant \(C\) has been improved \[63\], current techniques fall short of providing a super-linear lower bound. In fact, even the task of improving the constant in the basic case of 3 queries and binary responses requires highly non-trivial techniques \[1\]. (Stronger lower-bounds are known for some special cases, see \[1\] and \[67\] for references.)

- **Secret Sharing Schemes (SSS)** [58, 46]: In a secret-sharing scheme a dealer wishes to distribute a secret \(s \in \{0, 1\}\) among \(n\) computationally-unbounded servers such that only “authorized” coalitions of servers can recover the secret \(s\). That is, given \(s\) and a collection \(f \subset 2^{[n]}\) of authorized subsets, the dealer should randomly map \(s\) to \(n\) random shares \(s_1, \ldots, s_n\) such that any coalition \(x \in f\) can recover \(s\) based on their vector of shares \(s_x = (s_i)_{i \in x}\), and the shares \(s_x\) of any “unauthorized” coalition \(x \notin f\) leak no information on the secret \(s\). To make the problem feasible, we assume that \(f\) is monotone (i.e., a super-set of an authorized set is also authorized), and represent it as a monotone Boolean function \(f : \{0, 1\}^n \rightarrow \{0, 1\}\). The cost of SSS is measured as the maximal share-size \(\max|s_i|\), and determining the best achievable cost, for a worst-case function \(f : \{0, 1\}^n \rightarrow \{0, 1\}\), is considered to be a central open question in information-theoretic cryptography. (See the survey \[15\].) The best-known upper-bound is \((3/2)^{n+o(n)}\) \[9\] and the best-known lower-bound (from the mid 90s!) is \(\Omega(n / \log n)\) \[27, 28\].

- **Fully-Decomposable Randomized Encodings (DRE)**: Consider the problem of securely evaluating a public function \(f : \{0, 1\}^n \rightarrow \{0, 1\}\) where the input \(x = (x_1, \ldots, x_n)\) is distributed between \(n\) servers and the output should be delivered to a client without revealing the value of the inputs. We assume that each server holds a single input bit \(x_i\) and a common random string \(r\) that is hidden from the client. Each server is allowed to send a single message \(m_i = m_i(x_i, r)\), and based on the combination \((m_1, \ldots, m_n)\) of all these messages the client should be able to decode the value of \(f(x)\) without learning any additional information on \(x\). While the Randomized Encoding terminology is relatively modern \[8\], this notion goes back to the seminal works on Garbled

\[1\] One typically thinks of the truth table of \(f\) as an \(N\)-bit database for \(N = 2^n\). We use functional notation in order to unify the presentation.
Circuits [64] and on Private Simultaneous Message Protocols [34, 44]. (See the surveys [43, 2].) The cost of DRE is the maximal message length $\max_i |m_i|$. To the best of our knowledge, the best upper-bound is $2^{n/2+o(n)}$ [18, 19] and the best lower-bound is $\Omega(n/\log n)$ [13].

While all the above primitives share a simple communication pattern their semantics are very different. In DRE the information flows in one-way from servers to client, the function $f$ is public and privacy holds with respect to the servers’ inputs which are fully distributed. In SSS the information flow is from the client who holds a single secret bit to the servers who initially hold no input. The (public) function $f$ only determines which subsets can recover the client’s secret (so the function is always applied to public input). Finally, in PIR the information flows in both directions, the server’s data $f$ is duplicated, and only the client’s input to the (non-public) $f$ should be hidden. Nevertheless, as already observed the state of lower bounds in all these cases is somewhat similar and is extremely poor.

We emphasize that lower bounds are not known even for non-explicit functions. In this sense, the situation here is much worse than the lack of lower bounds in computational complexity. Furthermore, unlike the computational setting (both in cryptography and complexity) where we have a web of formal reductions between different problems/primitives that allows us to talk about complete tasks or focus on a concrete “hardest” problem, no such reductions are known in the current context. Indeed, we currently have only weak connections between some of the above primitives. As a result, a lower bound for, say secret sharing does not imply a lower bound against PIR, and vice versa. The main goal of this work is to partially remedy the situation by providing a single working hypothesis that implies strong lower bounds against all the above problems. Specifically, we relate all these questions to the classical topic of query/space trade-offs, lifted to the setting of interactive proof systems.

1.1 Advisor-Verifier-Prover Games

Consider the following game (hereafter referred to as AVP game) in which an Advisor and a Verifier wish to collaboratively certify that some function $f$ evaluates to 1 over some input $x$ via the aid of an untrusted Prover. The game is partitioned into an offline phase and an online phase. In the offline phase, an arbitrary function $f : \{0,1\}^n \rightarrow \{0,1\}$ is selected and is delivered only to the advisor who computes some secret (randomized) advice $a$. This advice is delivered privately to the verifier and the advisor leaves the room. Next, in the online phase, the verifier receives an input $x \in \{0,1\}^n$ and sends a single query $b = b(a,x)$ to an untrusted prover. The prover, who holds $x$ and $f$, tries to convince the verifier that $f(x) = 1$ by sending a single message $c = c(b,x,f)$. The verifier may accept or reject the answer by computing some predicate over $a,x,c$ and its private random tape. Importantly, the advice is hidden from the prover and we do not require any form of re-usability of the advice. That is, we measure soundness with respect to a single use of the system over a fresh advice. To keep the problem simple, we assume that all the parties are computationally-unbounded, and ask: How many bits of communication, $|a|+|b|+|c|$, are needed in order to derive such a protocol with, say perfect completeness and soundness error of $1/2$? Before tackling this question, it is instructive to compare AVPs to some previous, well-studied, models.

\hspace{1cm} Figure 1: The interaction pattern of AVP protocols.

\hspace{1cm} 2For example, some “nicely-structured” PIR lead to some variants of Randomized Encodings [18, 53] which, in turn, yield secret-sharing schemes with non-trivial exponential overhead $2^{\Omega(n)}$ for $C < 1$ [53, 51, 4, 5, 9, 6].
Comparison with other models. First, consider the case where there is no prover at all, i.e., the advisor who holds \( f \) has to send a single message to the verifier who holds \( x \) that allows her to determine with high probability if the outcome is 1 or not. This problem is essentially equivalent to the one-way randomized communication complexity for which a lower-bound of \( \Omega(2^n) \) was given in [49]. Next, consider the variant where the prover is trusted and is restricted to answer \( f \)-queries on inputs that are different from \( x \). This variant essentially corresponds to (the non-adaptive version of) Yao’s black-box model [55] for which a tight lower-bound of \( \Omega(2^{n/2}) \) is well-known. (More generally, the number of queries must be at least \( 2^n/|a| \).) Next, consider the setting where the advice should be reusable over an arbitrary number of invocations where in each invocation the verifier and the prover gets a different input \( x \). In this case, the problem is essentially equivalent to the notion of online (read-only) memory checking [23] [55]. Here too a tight trade-off of \( |c| \geq 2^n/|a| \) is proved by [55] which, again, implies an exponential lower-bound of \( 2^{n/2} \) on the total communication. If, on the other hand, we relax the AVP model and allow the prover and verifier to interact for multiple rounds (for a single instance), then we get the notion of non-uniform delegation as defined in [40 Section 4.3]. In this model, Goldwasser et al. show that when \( f \) is computed by a \( D \)-depth \( S \)-size circuit, it admits multi-round AVP protocol with communication of \( \text{poly}(D, \log(S)) \) and \( D \log n \) communication rounds. Since every function can be computed by \( O(n) \)-depth \( O(2^n) \)-size circuit this leads to a \( \text{poly}(n) \) communication in \( O(n \log n) \) rounds.

We observe that, for general functions, one can derive a \( \text{poly}(n) \) solution even when a single round of interaction is allowed as illustrated by the following example.

Example 1.1 (Arithmetization-based AVPs). Fix some finite field \( \mathbb{F} \) of size at least \( n+2 \) and let \( g : \mathbb{F}^n \rightarrow \mathbb{F} \) denote the (unique) multilinear extension of \( f : \{0,1\}^n \rightarrow \{0,1\} \) over the field \( \mathbb{F} \) (i.e., \( g(x) = f(x) \) for every 0-1 vector \( x \)). The advice \( a \) contains a random point \( x^* \in \mathbb{F}^n \) together with its evaluation \( y^* = g(x^*) \). In the online phase, the verifier, who holds \( x \) and \( a \), samples \( n+1 \) correlated points \( X = (x_1, \ldots, x_{n+1}) \) whose \( g \)-values determine \( f(x) \) and such that the known instance \( x^* \) lies in a random location in the tuple \( X \). The prover should send the \( g \)-evaluations of all the entries in \( X \) and cheating can be detected by verifying that the answer on \( x^* \) is consistent with the advice. Concretely, to generate the tuple \( X \), the verifier fixes some (public) tuple of \( n+1 \) non-zero distinct elements \( \alpha_1, \ldots, \alpha_{n+1} \in \mathbb{F} \), samples an index \( i \in [n+1] \) sets \( r = (x^* - x)/\alpha_i \in \mathbb{F}^n \) and generates the point \( x_j = x + r \cdot \alpha_j \) for each \( j \in [n+1] \). Given the prover’s answers \( y_1, \ldots, y_{n+1} \) (supposedly \( g(x_i) = y_i \)) the verifier verifies that \( y_i = y^* \), interpolates a (unique) degree-\( n \) univariate polynomial \( h \) for which \( h(\alpha_i) = y_i \) and accepts if \( h(0) = 1 \). It is not hard to verify that an honest prover will always convince the verifier and that a cheating prover will be caught with a probability of at least \( 1/(n+1) \). The error can be decreased to constant via \( O(n) \) parallel repetitions and the total communication complexity (after repetition) is \( O(n^3 \log |\mathbb{F}|) = O(n^3 \log n) \) bits. \( \triangle \)

In the above example, the prover’s answer is polynomially long in \( n \). Consider the case where the prover sends only \( C \) bits where \( C \) is a constant that does not grow with \( n \). We refer to such a prover as laconic and conjecture that in this case, either the advice or the query must be super-polynomially long. That is, the total communication complexity, \( |a| + |b| + |c| \) must be super-polynomial.

Hypothosis 1.2 (AVPs are Lengthy). Every prover-laconic AVP protocol that works for the class of \( n \)-bit predicates must have total communication complexity that is super polynomial in \( n \).

Back to Information-Theoretic Primitives. Hypothesis 1.2 allows us to derive super-polynomial lower bounds on all the above primitives.

Theorem 1.3 (main). Under Hypothesis 1.2, \( \text{sd-PIR over } 2^n \)-bit database, \( \text{SSS over } n \)-bit predicates, and \( \text{DRE over } n \)-bit predicates have all communication cost that grows super-polynomially in \( n \).

---

3One should note that in the one-way communication setting and Yao’s model the goal is to compute \( f \) whereas in our model the goal is to certify that \( f(x) = 1 \). This minor difference does not affect the asymptotic complexity since one can reduce computation to certification by concurrently running two copies of an AVP protocol once for \( f \) and once for \( -f \).
Thus, the AVP model can serve as a single hub for lower bounds: Any lower bound on its complexity would also imply similar lower bounds for PIR, SSS, and DRE. Hypothesis 1.2 can be used both as a working hypothesis or as an (ambitious) target for future works. As a first step, we show that one can adopt simple “counting-based” lower bounds to this model, and re-derive in a unified way some of the existing lower bounds that were previously proved separately for each model [47, 35, 50, 20]. (See Section 4.) Unfortunately, we do not get the best-known lower bounds for any of the above primitives. We will get back to this point later.

Theorem 1.3 is established by presenting communication-efficient transforms from each of the above primitives to a laconic-prover AVP protocol. As a by-product, these connections yield a non-trivial AVP with sub-exponential communication complexity of $2^\Theta(\sqrt{n})$ with a laconic prover who communicates a single bit! (See Section 3). In fact, as the reader may have noticed, the non-laconic AVP from Example 1.1 is also rooted in the PIR literature and can be derived from the n-server PIR protocol of [26].

Our transforms from IT primitives to AVPs are fairly simple. Indeed, previous works already established several relations between different variants of interactive proofs and secure computation protocols to randomized encodings [39, 38, 10, 11] to SSS/CDS [60] and to PIR protocols [42]. Our main conceptual contribution is the formalization of a single, arguably natural proof model, from which all the above primitives can be reduced.

**AVPs with Additional Features.** Notably, each primitive induces different additional properties over the resulting AVP, thus allowing a clean comparison between these different notions. Specifically we consider the following features:

1. **Simple AVPs.** We say that an AVP is simple if the verifier satisfies the following 2 properties:
   1. (Input Independent Verdict) The verifier’s final decision (accept/reject) depends only on the advice $a$ and the prover’s answer $c$ and is independent of the input $x$ or the verifier’s query; and
   2. (Decomposability) the verifier’s query $b$ can be partitioned to $n$ blocks $b_1, \ldots, b_n$ where $b_i = b_i(a, x_i)$ depends on the advice $a$ and on the $i$th bit of the input $x$. We note that “input-independent verdict” can be always guaranteed in the non-laconic setting (see Section 2.1), but it is unclear how to generically add it in the laconic setting.

2. **Monotonicity:** A simple AVP is monotone if the $i$th block of the verifier’s query $b_i$ is taken to be the empty string whenever the $i$th bit of the input $x_i$ is zero. Here “monotonicity” refers to the information that is given to the prover: whenever an input bit $x_i$ is flipped from 0 to 1 the prover only gets more information. Indeed, a simple monotone AVP protocol can be applied only to monotone functions. Nevertheless, it is not hard to show that monotone laconic-AVP with $\text{poly}(n)$ complexity for monotone functions implies laconic-AVP for general (non-monotone functions) with $\text{poly}(n)$ complexity as well (see Observation 3.2).

3. **Instance-Hiding.** An AVP is instance hiding if the verifier’s query leaks no information on the input $x$ and if an honest prover can answer the question solely based on $f$ without getting $x$ as an input. Soundness should still hold even if $x$ is given to the prover.

PIR protocols lead to plain AVP that fail to satisfy any of the above properties (including decomposability and Input Independent Verdict). In contrast, SSS and DRE both yield simple AVP with an extra property: monotonicity for SSS-based constructions, and instance-hiding for DRE-based AVPs. We also note that simple AVPs with no additional properties follow from another natural and well-studied information-theoretic primitive, Fully-Decomposable Conditional Disclosure of Secrets (CDS) [37] that is implied by both SSS and DRE. Roughly, this is a variant of secret sharing in which the parties are partitioned into pairs, and secrecy/correctness are defined only over coalitions that contain a single party per pair. (See Definition 2.4) Overall, the transformations to AVPs and the extra properties induce a new partial order among the primitives as depicted in Figure 2. For example, it suggests that, in a sense, PIR schemes should be “easier to achieve” than, say, DREs.
Figure 2: For primitives $P_1$ and $P_2$, a solid arrow $P_1 \rightarrow P_2$ indicates that a protocol for $P_1$ implies a protocol for $P_2$ with the same complexity. The dashed arrow $P_1 \rightarrow P_2$ indicates that there exists a separation between the primitives and $P_2$ is “stronger” than $P_1$ (the result we prove in Sections 5 and 6). The text near the arrows describes the properties of the AVP protocol implied by each of the primitives. Recall that Simple AVP is both decomposable and has input-dependent verdict.

1.2 Lifting Existing Lower-Bounds to AVPs?

While proving our main hypothesis currently seems beyond reach, it is natural to try to prove more modest lower bounds for AVP communication. The first step we suggest is to take existing lower bounds for SSS, DRE and PIR and to examine whether they can be generalized to bounds for AVP protocols. The known lower bounds for the aforementioned primitives were established with techniques from seemingly unrelated domains: information theory for SSS [28], formula lower bounds for DRE [13] and even quantum computation for PIR [62]. Is it possible to lift these bounds to the AVP setting and re-derive the best existing lower-bounds in a unified way?

On the positive side, we show in Section 4 that basic counting-based arguments can be generalized to AVPs and prove that if the prover’s computation is taken from some class $G$ then $\log G \cdot A$, the bit-length of the advice, must be $\Omega(2^n)$. This allows us to unify and extend several results that were previously proved separately for each primitive, including lower-bound for multilinear PIR [47], linear CDS protocol [35, 16], SSS with bounded receivers [50], and even general two-party CDS protocols [7, 12] as well as low-degree CDS protocols [20]. This also allows us to conclude that when in any AVP the advisor’s message length $A$, the verifier’s message length $B$ and the prover’s message length $C$ must satisfy the inequality

$$A \cdot 2^B \cdot C = \Omega(2^n).$$

For a more formal description see Corollary 4.5.

On the negative side, we show that some of the existing lower-bounds cannot be lifted to the AVP domain by proving new separations. Specifically, in Sections 5 and 6, we show that the best-existing lower bounds for SSS and for DRE do not generalize to AVP protocols. Roughly speaking, Csirmaz’s [28] lower-bound effectively defines a complexity measure $C$ over functions such that the secret-sharing cost of a function $f$ is lower-bounded by $C(f)$. We present a function $f$ for which $C(f)$ is $\Omega(n^2/\log n)$ but it admits a cheap CDS (and AVP) protocol of complexity $n^{1.5+\omega(1)}$. For partial functions, we can improve the CDS complexity to $n^{1+\omega(1)}$, deriving an almost quadratic gap. Similarly, Ball et al. [13] use Nechiporuk’s [32] complexity measure to lower-bound the DRE complexity of a function $f$. Here too, we present a function whose complexity under this measure is $\Omega(n^2/\log n)$ but admits an almost-linear $n^{1+\omega(1)}$-cost CDS (and therefore a cheap AVP). The driving force in both our constructions is the efficient (yet still with super-polynomial communication) CDS construction of [53] for general functions. Our constructions provide the first separations between fully-decomposable CDS to SSS and fully-decomposable CDS to DRE.

Our separations also highlight the fact that currently, we do not have any non-trivial lower bounds for (fully decomposable) CDS. Specifically, to the best of our knowledge, we do not even have a super-
constant lower-bound in this case. This seems to be a great target for future research. It is known that when the secret is huge it is possible to construct CDS schemes with constant-rate, and so natural lower-bound techniques that apply to the rate, are deemed to fail.

1.3 How plausible is Hypothesis 1.2?

It is natural to ask whether Hypothesis 1.2 can be somehow justified, especially in light of the existence of polynomial AVPs with non-laconic prover (Example 1.1). We provide some justification for the conjecture by proving a partial converse theorem for the case of regular AVPs, where regularity means that the number of prover’s answers that are accepted by the verifier is a fixed quantity independent of $f, x$ and the underlying randomness. Indeed, all known AVP protocols satisfy this assumption.

**Theorem 1.4** (partial converse). *If every $n$-bit predicate has a poly($n$)-bit regular AVP with a prover that communicates less than $c$ bits where $c$ is some universal constant, then the 2-party universal function $U_n : \{0,1\}^2 \times \{0,1\}^n \rightarrow \{0,1\}$ admits a statistical 2-party one-way communication CDS protocol with poly($n$) communication.*

Roughly speaking, in a 2-party one-way communication CDS protocol for the universal function, Alice holds a predicate $f : \{0,1\}^n \rightarrow \{0,1\}$ (represented as a string of length $2^n$), Bob holds a string $x \in \{0,1\}^n$ and a secret $s \in \{0,1\}$, Alice sends to Bob a short randomized message $a$ of length poly($n$) and Bob generates a short (poly($n$)-long) encoding $e$ that, together with $x$ and $f$, reveals the secret $s$ if and only if $f(x) = 1$. In fact, we will allow some error both in the correctness and in the privacy. This variant naturally extends the standard notion of 2-party CDS in which the encoding $e$ is generated non-interactively based on shared randomness $r$, i.e., $e = (e_1, e_2)$ where Alice samples $e_1 = e_1(f;r)$ and Bob samples $e_2 = e_2(x;r)$. (See [12, Section 9] for more general variants of interactive CDS).

The existence of CDS with polynomial complexity seems to be beyond current techniques, and, as far as we know, one-way communication from Alice (who holds the function) to Bob (who holds the evaluation point) does not seem to make the problem easier. Thus, Theorem 1.4 provides some evidence for Hypothesis 1.2.

Note that the theorem assumes that the prover is “very laconic”, i.e., she communicates less than $c$ bits for some universal constant $c$. While this assumption (as well as the regularity condition) can be somewhat relaxed (see Section 7), we currently do not know how to prove the theorem based on a polynomial AVP in which the prover’s communication complexity is bounded by an arbitrary constant. Still, the theorem highlights the qualitative difference between AVPs in which the prover’s communication complexity is large and AVPs in which the prover’s communication complexity is small. In particular, the non-laconic AVP construction from Example 1.1 fails to satisfy the required condition, whereas the constructions of AVPs from randomized encoding, SSS and CDS have a “very laconic” prover. It should be mentioned that the qualitative difference between laconic and non-laconic provers also arises in other contexts, e.g., [60][22].

**Organization.** In Section 2 we formally define AVP protocols, and the information-theoretic primitives used in this paper (SSS, DRE, CDS and PIR). Transformations from these primitives to AVPs are presented in Section 3. In Section 4 we prove the counting-based lower bounds for AVPs, and in Sections 5 and 6 prove the separations between CDS and secret sharing and between CDS and decomposable randomized encodings. Finally, the converse theorem appears in Section 7.

---

4We thank the anonymous reviewers of FOCS’23 for raising this question.
2 Preliminaries

2.1 AVPs

**Definition 2.1.** An AVP for a family $\mathcal{F}$ of $n$-bit predicates is a tuple $Q = (A, V, P, V_{acc})$ that satisfies the following properties.

1. **Syntax:** The advice algorithm $A$ takes a enumerate of a function $f \in \mathcal{F}$ and a random tape $r_A$ and outputs an advice string. The query algorithm $V$ takes as an input an advice $a$ and random $x$; and a random tape $r_V$, and generates a query. The response function $P$ takes a query $b$ an input $x$ and a function $f$ and generates a response, and the predicate $V_{acc}$ takes a verifier’s view $(a, x, r_V, c)$ and decides whether to accept or reject it.

2. **Perfect completeness:** For every $f \in \mathcal{F}$ and $x$ such that $f(x) = 1$ it holds that
   \[
   \Pr_{r_A,r_V} [V_{acc}(a, x, r_V, c) = 1] = 1,
   \]
   where $a = A(f; r_A), b = V(a, x; r_V)$ and $c = P(b, x, f)$.

3. **Soundness:** For every cheating prover’s strategy $P^*$, every $f \in \mathcal{F}$ and every $x$ such that $f(x) = 0$, it holds that
   \[
   \Pr_{r_A,r_V} [V_{acc}(a, x, r_V, c^*) = 1] \leq 0.5,
   \]
   where $a = A(f; r_A), b = V(a, x; r_V)$ and $c^* = P^*(b, x, f)$.

We say that $Q$ realizes a function $f$ with communication $R_Q$ if the protocol messages when $f$ is taken as an input satisfy $\max_{x,a,r_V} [A(f; r_A)] + |V(a, x; r_V)| + |P(b, x, f)| \leq R_Q$. We say that the protocol has a communication complexity $R_Q$ if $R_{f,Q} \leq R_Q$ for every $f \in \mathcal{F}$. By default, we take $\mathcal{F}$ to be the class of all $n$-bit predicates, and measure the complexity $R_Q$ as a function of $n$.

We say that the prover is laconic if $P$ always outputs a constant number of bits that does not grow with $n$. Of special interest will be the case of a 1-bit prover that communicates a single bit.

Few comments are in place. First, the notion of AVP (with unbounded verifiers) is meaningless when restricted to a single function $f$ since there exists an AVP protocol with no communication for every fixed $f$ (the function can be hard-wired into the verifier). This problem vanishes if we put some restriction on the verifier (e.g., restrict its computational complexity). We also note that the soundness error can be amplified by parallel repetition and so the choice of $1/2$ is somewhat arbitrary. Also, one can always assume that the verifier is deterministic by embedding her coins as part of the advice. By using standard randomness sparsification, this does not increase the total amount of communication by too much (see Lemma [A.3]). Finally, we emphasize that the definition does not require any form of re-usability of the advice.

Additional properties. The protocol has input-independent verdict if $V_{acc}$ depends only in $a$ and $c$. The protocol is decomposable if $b = V(a, x; r_V)$ can be written as $(b_1(a, x_1), \ldots, b_n(a, x_n))$ where $x_i$ is the $i$th bit of $x$. A decomposable protocol with input-independent verdict is simple. A simple protocol is monotone if for every $i \in [n]$, it holds that $b_i(a, 0, r_V) = \perp$ for every $a, r_V$. The protocol is instance hiding if (1) for every $f$ and every $x, x'$ the random variables $b = V(a(f; r_A), x; r_V)$ and $b' = V(a(f; r_A), x'; r_V)$, induced by a random choice of $r_A$ and $r_V$, are identically distributed; and (2) if the response function $P(b, x, f)$ depends only on $b$ and $f$ and is independent of $x$.

We note that in the non-laconic setting, one can move from input-dependent to input-independent acceptance easily. The advisor samples a key $k$ of a message authentication code (e.g., a pairwise independent hash function) and appends it to its advice, the verifier sends the tags $t = H_k(x)$ to the prover as part of his query and the prover appends to its answer the pair $(x, t)$. The verifier rejects if $H_k(x) \neq t$, and otherwise, runs the original test over the received $x$. Note that this increases the communication from the prover to the verifier by $O(n)$ bits and so the resulting prover is not laconic.
2.2 Secret-Sharing Schemes

For a string \( x \in \{0,1\}^n \) we let \( I_x \) be the set \( \{i : x_i = 1\} \subset [n] \) and write \( x \leq x' \) if \( I_x \subseteq I_{x'} \). A predicate \( f : \{0,1\}^n \to \{0,1\} \) is monotone if for every \( x \leq x' \) it holds that \( f(x) \leq f(x') \). We present the definition of secret-sharing schemes, similar to [14] [25]. For the privacy of these schemes, we use the following notation: For two random variables \( X \) and \( Y \), we say that \( X \equiv Y \) if they are identically distributed.

**Definition 2.2** (Secret-sharing schemes). A secret-sharing scheme, with domain of secrets \( S \), domain of random strings \( R \), and finite domains of shares \( S_1, \ldots, S_n \), is a deterministic function \( D : S \times R \to S_1 \times \cdots \times S_n \). A dealer distributes a secret \( s \in S \) according to \( D \) by first sampling a random string \( r \in R \) with uniform distribution, computing a vector of shares \( D(s,r) = (s_1, \ldots, s_n) \), and privately communicating each share \( s_i \) to the \( i \)-th party. For a binary string \( x \in \{0,1\}^n \) representing a set \( I_x = \{i : x_i = 1\} \), we denote \( D_x(s,r) \) as the restriction of \( D(s,r) \) to the \( I_x \)-entries (i.e., the shares of the parties in \( I_x \)). A secret-sharing scheme \( D \) realizes a monotone predicate \( f : \{0,1\}^n \to \{0,1\} \) if the following two requirements hold:

1. **Perfect Correctness:** The secret \( s \) can be reconstructed by any authorized set of parties. That is, for every \( x \) such that \( f(x) = 1 \) there exists a reconstruction function \( \text{Recon}_x \) such that for every secret \( s \in S \) and every random string \( r \in R \), it holds that \( \text{Recon}_x(D(s,r)) = s \).

2. **Perfect Privacy:** Any unauthorized set cannot learn anything about the secret from its shares. Formally, for every \( x \) such that \( f(x) = 0 \) and every pair of secrets \( s,s' \in S \), it holds that \( D_x(s,r) \equiv D_x(s',r) \), where \( r \) is sampled with uniform distribution from \( R \).

The secret size in a secret-sharing scheme \( D \) is defined as \( \log |S| \) and the share size of the scheme \( D \) is defined as the largest share size, i.e., \( \max_{1 \leq i \leq n} \log |S_i| \). The scheme \( D \) is a linear secret-sharing scheme over a finite field \( \mathbb{F} \) if \( S = \mathbb{F} \), \( R = \mathbb{F}^d \) for some integer \( d \geq 1 \), the sets \( S_1, \ldots, S_n \) are vector spaces over \( \mathbb{F} \), and the function \( D : \mathbb{F}^{d+1} \to S_1 \times \cdots \times S_n \) is a linear mapping over \( \mathbb{F} \). By default, linearity is defined over the binary field \( \mathbb{F}_2 \).

Here and throughout the paper we assume, for simplicity, perfect privacy, and perfect correctness. All of our reductions extend easily to the setting of statistically private and statistically correct primitives. The secret-sharing definition also generalizes to partial monotone functions over a subset \( X \) of \( \{0,1\}^n \). In this case, privacy and correctness are defined over inputs in \( X \) for which \( f(x) = 0 \) and \( f(x) = 1 \), respectively.

2.3 Fully Decomposable Randomized Encoding

**Definition 2.3** (Randomized encoding [45] [8]). Let \( f : X \to Y \) be some function. We say that \( \text{ENC} : X \times R \to Z \) is a randomized encoding (RE) of \( f \) if the following conditions hold:

1. **Perfect correctness:** There exists a deterministic reconstruction function \( \text{Dec} \) such that for every \( x \in X \) and every random string \( r \in R \) it holds that \( \text{Dec}(\text{ENC}(x;r)) = f(x) \).

2. **Perfect Privacy:** There exists a randomized function \( \text{SIM} \), called a simulator, such that for every input \( x \in X \), the distribution \( \text{SIM}(f(x)) \) is identically distributed to the distribution \( \text{ENC}(x;r) \) induced by uniformly sampling \( r \in R \).

We say that a RE is fully-decomposable (DRE) if the input domain is \( X = \{0,1\}^n \) and if the encoding function \( \text{ENC} \) can be decomposed into \( n \) functions \( \text{ENC}_i : \{0,1\} \times R \to Z_i \) such that \( \text{ENC}(x_1, \ldots, x_n;r) = (\text{ENC}_i(x_i;r))_{i \in [n]} \) for every \( (x_1, \ldots, x_n) \in \{0,1\}^n \) and \( r \). By default, we will assume that the output domain \( Y \) of \( f \) is a single bit. The total size of DRE is the bit length of the output of the encoder, i.e., \( \log |Z| \), and the message complexity of the scheme is \( \max_i \log |Z_i| \).

---

5The share size is sometimes defined to be the total share size, i.e., \( \sum_{1 \leq i \leq n} \log |S_i| \). However, since the two differ by at most a linear factor of \( n \), the difference is not important in our context.
2.4 Conditional Disclosure of Secrets Protocol

Definition 2.4 (Conditional disclosure of secrets protocols [37]). Let \( f : X \to Y \) be some function. We say that \( \text{ENC} : X \times S \times R \to Z \) is a Conditional disclosure of secrets (CDS) of \( f \) with secret domain \( S \) if the following conditions hold:

1. \( \varepsilon \)-correctness: There exists a deterministic reconstruction function \( \text{DEC} \) such that for every \( x \in X \) such that \( f(x) = 1 \) it holds that \( \Pr_{r \in R}[\text{DEC}(x, \text{ENC}(x, s; r)) \neq s] \leq \varepsilon \).

2. \( \delta \)-privacy of secrets: For every input \( x \in X \) such that \( f(x) = 0 \) and every pair of secret \( s, s' \in S \) the distributions \( \text{ENC}(x, s, r) \) and \( \text{ENC}(x, s', r) \), induced by sampling \( r \) uniformly from \( R \), are \( \delta \)-close in statistical distance.

Throughout the paper (except for the results in Section 7), we focus on the case of perfect correctness and perfect privacy where \( \varepsilon = \delta = 0 \). For \( X = X_1 \times \cdots \times X_k \), we say that the CDS is a \( k \)-server CDS if the encoding function \( \text{ENC} \) can be decomposed into \( k \) functions \( \text{ENC}_i : X_i \times S \times R \to Z_i \) such that \( \text{ENC}((x_1, \ldots, x_k), s, r) = (\text{ENC}_i(x_i, s, r))_{i \in [k]} \) for every \( (x_1, \ldots, x_k) \in X_1 \times \cdots \times X_k \). This corresponds to a setting where there are \( k \) servers with private inputs \( x_1, \ldots, x_k \) respectively, and joint input \( (s, r) \) and where the \( i \)th server computes \( \text{ENC}_i(x_i, s, r) \). When \( X_i = \{0, 1\} \) for every \( i \in [k] \), we say that the CDS is fully-decomposable – this will be our default setting, and in this case we typically use \( n \) to denote the number of servers. The total size of CDS is the bit length of the output of the encoder, i.e., \( \log|Z| \), and the message complexity of the scheme is \( \max_i \log|Z_i| \). The CDS is linear over a finite field \( \mathbb{F} \) if \( S = \mathbb{F} \), \( R = \mathbb{F}^\ell \) for some integer \( \ell \geq 1 \), \( Z_1, \ldots, Z_n \) are vector spaces over \( \mathbb{F} \), and the function \( \text{ENC}_i : \mathbb{F}^\ell \to \mathbb{F} \) is a linear function over \( \mathbb{F} \) for every \( i \in [n] \). By default, we take \( \mathbb{F} \) to be the binary field \( \mathbb{F}_2 \).

Conditional disclosure of secrets can be viewed as a relaxation of both DRE and secret sharing.

Remark 2.5 (CDS vs Secret-Sharing). A fully-decomposable CDS for \( f : \{0, 1\}^n \to \{0, 1\} \) is nothing but a secret-sharing scheme for the partial function \( g : \{0, 1\}^n \to \{0, 1\} \) that is defined over the set of inputs \( \{(01), (10)\}^n \) and where \( g(y_1, \ldots, y_{2n}) = f(x_1, \ldots, x_n) \) where for every \( i \in [n] \), the pair \( (y_{2i-1}, y_{2i}) \) equals to 10 if \( x_i = 1 \) and to 01 if \( x_i = 0 \). Note that \( g \) is monotone in a vacuous way since it is defined only over inputs \( y \) of Hamming weight \( n \). (One can think of the secret-sharing parties as partitioned into pairs and we consider only coalitions that consist exactly a single participant from each pair.) Indeed, fully-decomposable CDS is sometimes viewed as non-monotone secret-sharing [17] [53]. Thus, general SSS implies general CDS with similar complexity.

Remark 2.6 (CDS vs DRE). A fully-decomposable CDS for \( f : \{0, 1\}^n \to \{0, 1\} \) can be obtained from a DRE for the predicate \( g : \{0, 1\}^{n+1} \to \{0, 1\} \) that takes a secret \( s \in \{0, 1\} \) and an \( n \)-bit input \( x \in \{0, 1\} \) and outputs \( s \) if and only if \( f(x) = 1 \). That is, \( g(x, s) = f(x) \land s \). In fact, the resulting DRE yields a CDS with input hiding property: The encoding hides the input \( x \), i.e., privacy is strengthened so that

\[
\text{ENC}(x, s, r) \equiv \text{ENC}(x', s', r),
\]

for every pair \( (x, s) \) and \( (x', s') \) for which \( g(x, s) = g(x', s') \), and for correctness the decoder needs only the encoding and does not need \( x \) as an additional input.

2.5 Private Information Retrieval

A \( k \)-server PIR protocol involves \( k \) servers \( S_1, \ldots, S_k \), each holding the same \( N \)-bit string \( y \) (the database), and a user who wants to retrieve a bit \( y_i \) of the data. PIR was originally defined by Chor et al. [26]. The definition is taken from [17].

Definition 2.7 (Private Information Retrieval [26] [17]). A \( k \)-server PIR protocol \((\mathcal{D}, \mathcal{U}_1, \ldots, \mathcal{U}_k, \mathcal{R}_1, \ldots, \mathcal{R}_k, \mathcal{C}) \) consists of a probability distribution \( \mathcal{D} \) and three types of algorithms: the user’s query algorithms \( \mathcal{U}_j(\cdot, \cdot) \), response algorithms \( \mathcal{R}_j(\cdot, \cdot) \), and a reconstruction algorithm \( \mathcal{C}(\cdot, \cdot, \ldots, \cdot) \) (\( \mathcal{C} \) has \( k+2 \) arguments).
At the beginning of the protocol, the user picks a random string $r$ from the distribution $\mathcal{D}$. For $j = 1, \ldots, k$, it computes a query $q_j = U_j(i, r)$ and sends it to server $S_j$. Each server $S_j$ responds with an answer $a_j = R_j(q_j, y)$. Finally, the user computes the bit $y_i$ by applying the reconstruction algorithm $C(i, r, a_1, \ldots, a_k)$. We assume that all algorithms are computationally unbounded. We also assume, without loss of generality, that both the response and reconstruction algorithms are deterministic. A PIR protocol is correct and secure if the following requirements hold:

1. **Perfect correctness**: The user always computes the correct value of $y_i$. Formally, for every $i \in \{1, \ldots, N\}$, every random string $r$, and every database $y \in \{0, 1\}^N$, $C(i, r, R_1(U_1(i, r), y), \ldots, R_k(U_k(i, r), y)) = y_i$.

2. **Perfect Privacy**: Each server has no information about the bit that the user tries to retrieve. Formally, for every two indices $i_1, i_2 \in [N]$ and every query algorithm $j \in [k]$, the distributions $U_j(i_1, r)$ and $U_j(i_2, r)$ induced by choosing $r \in \mathcal{D}$, are identical.

### 3 AVP Protocols and Information-Theoretic Primitives

#### 3.1 AVPs from Secret Sharing Schemes

We start by describing the connection between AVP protocols and secret sharing schemes.

**Theorem 3.1** (AVPs and secret sharing). If there exists a secret sharing scheme that for every monotone function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ has total share size $L$ then there exists a simple AVP protocol for monotone functions with perfect correctness, soundness $1/2$, and where the advice is of length $L + 1$, the verifier’s query is of length $L$ and the prover’s answer consists of 1 bit.

**Proof.** We describe the protocol. The advisor picks uniformly at random a bit $s$ and uses the given scheme to share it according to $f$ to shares $(s_1, \ldots, s_n)$ of total size $L$. It then sends the verifier the vector $a = (s, s_1, \ldots, s_n)$. Next, the verifier sends the prover all of the shares $s_i$ such that $x_i = 1$. Then if $f(x) = 1$ an honest prover can recover the secret $s$ and send it as its message $c$. Finally, the verifier outputs 1 iff $c = s$. Correctness is trivial and soundness follows from secrecy. Indeed, any cheating prover that convinces the verifier to accept with probability greater than 1/2 an input $x$ for which $f(x) = 0$, violates the perfect privacy of the secret sharing scheme.

By plugging in the best-known SSS [9] for general monotone functions, Theorem 3.1 implies that for every monotone function there exists an AVP protocol with complexity $1.5n + o(n)$. We further note that AVPs for monotone functions can be extended to AVPs for general functions.

**Observation 3.2.** If there exists an AVP for the class of all $n$-bit monotone predicates with communication complexity $R(n)$ then there exists an AVP for the class of all $n$-bit general predicates with communication complexity $R(2n)$. Furthermore, the transformation preserves simplicity and soundness error.

**Proof.** The parties map a non-monotone predicate $f : \{0, 1\}^n \rightarrow \{0, 1\}$ into the monotone predicate $g(x_1, x_1', \ldots, x_n, x_n')$ that agrees with $f(x_1, \ldots, x_n)$ whenever $x_i' = 1 - x_i$ and map an input $x$ for $f$ to the $2n$-bit input $(x_i, 1 - x_i)^n$ for $g$. Then use the monotone AVP over these inputs.

Since the verifier in the protocol based on secret sharing is monotone, even lower bounds for AVP protocols with monotone verifiers will imply lower bounds for the share size of secret sharing schemes.

**Remark 3.3 (about monotonicity).** As part of the monotonicity definition we put a simplicity requirement, i.e., decomposability and input-independence verdict. We note that these additional conditions are important. For decomposability, this is clear as monotonicity is not well-defined for non-decomposable protocols. For input-independence this may be less obvious. However, we note that if one does not put any restriction on the acceptance predicate, then “monotone” protocols can be applied to non-monotone
functions. To see this consider the following SSS-based AVP protocol. Given a non-monotone $f$ the prover defines $f_k$ to be the slice function that agrees with $f$ on $k$-weight inputs and outputs 0 and 1 on lighter and heavier inputs, respectively. The advisor shares for each $k \in [n]$ a random secret $s_k$ according to the monotone function $f_k$ and appends the shares and secrets as part of the advice $a$. Given $x$, the verifier sends all the shares that are indexed by $x$. At the end, the verifier accepts the prover’s message if it equals to the secret $s_i$ where $i$ is the Hamming-weight of $x$. It is not hard to verify that the protocol is correct, sound, decomposable and monotone.\footnote{In fact, since secret sharing for slice functions can be achieved with complexity of $2^{O(\sqrt{n})}$, the resulting protocol has sub-exponential complexity. We will get a similar result directly via CDS in the next section.} Getting back to our monotonicity definition, we note that one can replace the “input-independence verdict” property with the more liberal requirement that the acceptance predicate is monotone with respect to the input $x$. Indeed, under this definition monotone AVPs can be applied only to monotone functions.

3.2 AVPs from CDS and DRE

We can transform CDS to a simple (non-monotone) AVP in a way that is similar to the proof of Theorem 3.4.

**Theorem 3.4 (AVPs from CDS).** If there exists a fully-decomposable CDS for all predicates $f : \{0,1\}^n \to \{0,1\}$ with message size $L$ then there exists a simple AVP protocol for every function $f$ with perfect correctness, soundness $1/2$, and where the advice is of length $2nL + 1$, the verifier’s query is of length $nL$ and the prover’s answer consists of 1 bit. Furthermore, if the CDS is input-private (as per Remark 2.6) the AVP is instance-hiding.

**Proof.** The advisor samples a secret $s$ and randomness $r$ and generates, for each $i \in [n]$ a pair of CDS messages $\text{ENC}_i((0,s);r), \text{ENC}_i((1,s);r)$ the tuple $(\text{ENC}_i((0,s);r), \text{ENC}_i((1,s);r))_{i\in [n]}$ is given as part of the advice together with the secret $s$. Given $x$, the verifier sends to the prover the tuple $(\text{ENC}_i(x_i,s;r))_{i\in [n]}$ and receives back a bit $c$. The verifier accepts if $c = s$. Correctness is trivial, and soundness follows from the privacy of the CDS. Instance-hiding follows from input-privacy by definition.

Since there are fully-decomposable CDS protocols with message complexity of $2^{O(\sqrt{n})}$, we derive simple (non-monotone) AVPs with similar complexity.

**Remark 3.5 (using 2-server CDS).** Theorem 3.4 naturally extends to the 2-party setting. For concreteness, assume that we parse the input $x \in \{0,1\}^n$ into two equal halfs, $(x_1, x_2) \in \{0,1\}^{n/2} \times \{0,1\}^{n/2}$, and assume that the CDS encoding function $\text{ENC}(x,s;r)$ decomposes into $\text{ENC}_1(x_1,s;r)$ and $\text{ENC}_2(x_2,s;r)$. Then, the advisor can set her advice to be a secret $s$ together with the values $(\text{ENC}_1(x_1,s;r))_{x_1 \in \{0,1\}^{n/2}}$ and $(\text{ENC}_2(x_2,s;r))_{x_2 \in \{0,1\}^{n/2}}$. The verifier can select the message that corresponds to her input. This leads to an AVP with advice length of $2 \cdot 2^{n/2} \cdot L + 1$, verifier’s message length of $2L$, and prover’s message length of 1 bit.

**AVPs from DRE.** Our conversion of DRE to AVPs follows by constructing input-private CDS as explained in Remark 2.6 and by applying Theorem 3.4. We conclude the following theorem.

**Theorem 3.6 (AVPs from DRE).** If there exists a fully-decomposable DRE for all predicates $g : \{0,1\}^{n+1} \to \{0,1\}$ with message size $L$ then there exists a simple instance-hiding AVP protocol for every predicate $f : \{0,1\}^n \to \{0,1\}$ with perfect correctness, soundness $1/2$, and where the advice is of length $2(n+1)L$, the verifier’s query is of length $nL$ and the prover’s answer consists of 1 bit.

3.3 AVPs and Private Information Retrieval

We formalize the connection between AVPs and PIR protocols by the following theorem:
Theorem 3.7 (AVPs and PIR protocols). If there exists a $k$-server PIR protocol for databases of size $N = 2^n$ where the query for each server is of bit-length $L$ and the answer is of bit-length $T$, then there exists an AVP protocol for $n$-bit predicates with perfect correctness, soundness error of $(1 - 1/k)$ and where the advice is of bit-length $\log k + L + T$, the verifier’s query is of length $kL$ and the prover’s answer is of bit-length $kT$.

Proof. The advisor takes as input a function $f$ and views its truth-table as an $N$-bit database, she samples an index $i \in [k]$, generates a random pair of query/response $(q_i, \alpha_i)$ and sends the advice $a := (i, q_i, \alpha_i)$.

The query $q_i$ is sampled by invoking the client’s query algorithm on a fresh randomness and some fixed index $x^*$, and $\alpha_i$ is computed by applying the response algorithm of the $i$th server on the query $q_i$ with respect to the database $f$. The verifier takes the input $x \in \{0, 1\}^n$, and given the query $q_i$ received from the advisor, he samples randomness $r$ such that $q_i = U_i(x, r)$. Such a random string has to exist due to the privacy of PIR protocols: Any single query does not reveal information about the queried index, and thus every query must belong to the support of queries for all indices. The verifier then computes queries for the other $k - 1$ servers based on $r$ and generates for every $j \in [k]/\{i\}$ the query $q_j = U_j(x, r)$. The verifier sends to the prover $P$ the tuple of queries

$$b := (q_1, \ldots, q_k).$$

The prover: for every $j \in [k]$, the honest prover computes the answer $\alpha'_j$ of the $j$th PIR server over the query $q_j$ with respect to the database $f$, and sends to the verifier the answers

$$c := (\alpha'_1, \ldots, \alpha'_k).$$

Verifier output: To settle the verdict the verifier computes the reconstruction function of the PIR protocol $\gamma = C(x, r, \alpha'_1, \ldots, \alpha'_k)$, and accepts if and only if $\alpha_i = \alpha'_i$ and $\gamma = 1$.

Analysis: If $f(x) = 1$ it is clear that the verifier will always output 1 when interacting with an honest prover. When $f(x) = 0$, the verifier outputs 1 only if the prover changes at least one of the PIR response messages in a way that makes $C$ output 1. Since the server-index $i$ is not known to the prover and is chosen uniformly at random by the advisor, the probability that a malicious prover will be able to leave the $i$th message unchanged and cheat the verifier is at most $1 - 1/k$.

We note that in the AVPs based on SSS, DRE and CDS the prover’s response consists of a single bit, whereas in PIR-based AVPs the prover’s response is longer (at least 2 bits for the case of 2-server PIR with binary answers). The theorem can be applied even when the number of servers $k$ and the answer’s bit-length $L$ grow with $n$. In this case, we get an AVP with a non-laconic prover. Indeed, by applying Theorem 3.7 to the Reed-Muller based PIR of [26] that works with $\Theta(n)$ servers, we get the AVP protocol that appears in Example 1.1.

4 AVP Lower Bounds

4.1 Lower Bound on the Size of the Advice

While proving Hypothesis 1.2 seems to be out of reach given current techniques, it is possible to obtain several weaker lower bounds. We begin by lower-bounding the size of the advice. Through this section we let $\mathcal{F}$ denote the class of all $n$-bit predicates.

Theorem 4.1 (Lower bound on the size of the advice). AVP protocols that realize every function in $\mathcal{F}$ with perfect correctness and soundness error $\delta < 1$ require advice strings of size $n - 1$. 

13
This lower bound implies a similar bound for the communication in PIR, CDS, SSS and DRE, but for all these primitives such a bound is trivial. However, for PIR and CDS even an improvement by a constant factor will be a breakthrough. For PIR the best-known lower bound is $5n$ for the total communication complexity [62], and for fully-decomposable CDS we do not know any argument that excludes the possibility of total message size of even $3n$.

**Proof.** We begin with the following simple claim.

**Claim 4.2.** Let $Q = (A, V, P)$ be an AVP protocol with perfect correctness and a constant soundness error $\delta < 1$. Then for every pair of functions $f \neq f' \in \mathcal{F}$ the supports of the distributions of advisor-messages must be different.

**Proof of Claim 4.2.** Let $f \neq f' \in \mathcal{F}$ be functions that differ on an input $x$, and say without loss of generality that $f(x) = 1$ and $f'(x) = 0$. Assume towards contradiction that the supports of the distributions of advice messages of $Q$ is the same for both functions. Then the set of verifier messages for both functions on the input $x$ will also be identical. Denote this set as $B_x$. Then since $f(x) = 1$ and $Q$ has perfect correctness the prover $P$ can always persuade the verifier that $x$ is a 1-input of the $f$. It is then easy to describe a prover $P'$ that can cheat $A$ and $V$ on the function $f'$ and the input $x$. For every verifier message $b \in B_x$ define $P'(f', x, b) = P(f, x, b)$. The view of the verifier when communicating with $P'$ over the function $f'$ will be identical to its view when communicating with $P$ over the function $f$, and so it will always decide that $f'(x) = 1$, in contradiction to the existence of some non-trivial soundness for $Q$.

We can now prove a lower bound on the size of the advice by a counting argument. Suppose, towards a contradiction, that there exists an AVP protocol $Q = (A, V, P)$ whose advice is of size $M = n - 2$ for every function in $\mathcal{F}$. Denote by $\mathcal{M}$ the set of all strings of size $\leq M$. There are $\sum_{i=0}^{M} 2^i \leq 2^{M+1}$ such strings. Denote by $2^M$ the power set of $\mathcal{M}$, whose size is at most $2^{2M+1}$. For every function $f$ the support of the distribution of advice messages sent by the advisor when $f$ is taken as an input is an element in $2^\mathcal{M}$, and hence by the above claim the protocol can realize is at most $|2^\mathcal{M}| \leq 2^{2M+1} < 2^{2n}$ functions, the theorem follows.

### 4.2 A General AVP Lower Bound

Next, we show a bound based on counting arguments that generalize the ideas of [50]. They study the case where the secret sharing reconstruction can only be performed by a family of functions $\mathcal{G}_{Rec}$. They prove that for most monotone functions $f : \{0, 1\}^n \rightarrow \{0, 1\}$ the total share size $S$ of any scheme for $f$ must satisfy

$$S \cdot \log |\mathcal{G}_{Rec}| = \Omega(2^n / \sqrt{n}).$$

These ideas were generalized in [20], and we show that they actually hold in a more general setting of AVPs, which implies both [50] and [20] as well as prior bounds on linear-CDS [35] and multilinear PIR [47]. Our proof closely follows the presentation of [20].

For a function family $\mathcal{G}$, we say that a prover of an AVP protocol is $\mathcal{G}$-compatible if for every fixed $x$ and $f$, the honest prover answers the verifier’s query by computing a function $g$ from $\mathcal{G}$. (The choice of which $g$ to compute may depend on $f$ and $x$.) For the sake of lower bounds we consider AVPs with imperfect correctness (this only strengthens the statement).

**Theorem 4.3** (AVP lower bounds for families of provers). Let $Q$ be an AVP protocol for all functions in $\mathcal{F}$ with correctness and soundness errors $1/4$ with a $\mathcal{G}$-compatible prover and deterministic verifier.\footnote{We focus on deterministic verifiers for simplicity and since all our transformations from SSS, CDS, DRE and PIR yield deterministic verifiers. However, this requirement can be easily waived with minor additional cost by using the sparsification lemma for the verifier’s randomness (Lemma 5.3) and pushing the randomness as part of the advisor’s advice.}
Then for almost all functions \( f \in \mathcal{F} \) the lengths of the advisor’s message denoted by \( A_f \) satisfy
\[
A_f \cdot \log |\mathcal{G}| = \Omega(2^n).
\]

The proof of Theorem 4.3 is postponed to Section 4.5.

**Remark 4.4** (extension to general function families). We mention that the theorem holds for any class of Boolean functions \( \mathcal{F}' \) over some domain \( X' \) at the expense of replacing the \( \Omega(2^n) \) bound with \( \Omega(\mathcal{VC}(\mathcal{F}')) \) where \( \mathcal{VC}(\mathcal{F}') \) is the Vapnik–Chervonenkis (VC) dimension \([61]\) of \( \mathcal{F}' \). This follows by noting that an AVP for \( \mathcal{F}' \) can be converted into an AVP for the class \( \mathcal{F}_n \) of all Boolean functions over strings of length \( n = \log(\mathcal{VC}(\mathcal{F}')) \). Indeed, this can be done by defining an injective mapping \( \alpha \) from \( \mathcal{F}_n \) to \( \mathcal{F}' \) and an injective mapping \( \beta \) between \( \{0, 1\}^n \) to \( X' \) such that \( f(x) = \alpha(f)(\beta(x)) \) for every \( f \in \mathcal{F}_n \) and \( x \in \{0, 1\}^n \). The advisor, prover and verifier can apply these mappings locally and use an \( \mathcal{F}' \)-AVP to derive a \( \mathcal{F}_n \)-AVP. By plugging in the AVP lower bound for \( \mathcal{F}_n \), we derive the lower bound for \( \mathcal{F}' \). A similar observation is implicit in the proof of [20]. This extension will not be used in this paper, and is recorded for future works.

Let us apply Theorem 4.3 to several families of prover functions, starting with the case of an unrestricted prover that can send the verifier the result of any computation from \( \leq B \) bits to \( \leq C \) bits. There are \( \leq (2^{C+1})^{(2^B+1)} \) such functions, and so the following corollary follows:

**Corollary 4.5** (AVP lower bound for general provers). Let \( Q \) be an AVP protocol for all functions in \( \mathcal{F} \) with a deterministic verifier and with perfect correctness and soundness errors 1/4. Denote for every \( f \in \mathcal{F} \) by \( A_f \) the length of the advisor message of \( Q \), and the maximal length of the verifier’s and prover’s messages by \( B \) and \( C \). Then for almost all functions \( f \in \mathcal{F} \) it holds that
\[
A_f \cdot 2^{B+1} \cdot (C + 1) = \Omega(2^n).
\]

In particular, when both the advisor and prover send sub-exponential messages, i.e., \( A, C = 2^{o(n)} \), the message sent by the verifier has to be of length at least \( n - O(1) \). When the verifier is “silent”, i.e., when \( B = 0 \), we get that for most functions \( A_f \cdot C = \Omega(2^n) \), and that the total communication must be at least \( \Omega(2^{n/2}) \). In Appendix A.1 we show that this bound is tight by constructing a silent-verifier AVP with \( O(2^{n/2}) \) complexity mimicking similar results from the literature on Yao’s black-box model and program checking [65] [55].

**Remark 4.6** (lower bound for general 2-server CDS). Corollary 4.5 also yields an \( \Omega(n) \) lower on the message complexity \( L \) of 2-server CDS. Indeed, by Remark 3.5 such a CDS leads to an AVP with advice \( A = O(2^{n/2}L) \), verifier’s message of length \( B = O(L) \) and 1-bit prover’s message, and so, by Corollary 4.3 we get that \( L = \Omega(n) \). This asymptotically matches the best known lower bounds for 2-server CDS from [7][12].

### 4.3 Arithmetic Low-Degree Provers

Assume that the verifier communicates \( B' \) elements of a finite field \( \mathbb{F} \) (i.e., the number of bits is \( B = B' \log|\mathbb{F}| \)) and that the prover responds with \( C' \) field elements (i.e., communicates \( C = C' \log|\mathbb{F}| \) bits) and that, for every \( f \) and \( x \), the prover’s computation can be written as a degree-\( d \) mapping. Since the number of such mappings \( \mathcal{G} \) is at most \( |\mathbb{F}|^{C'(B'+d)/d} \) it holds that
\[
\log |\mathcal{G}| \leq C' \left( \frac{B'+d}{d} \right) \log |\mathbb{F}| \leq C(B/\log|\mathbb{F}|+1)^d \leq C(2B/\log|\mathbb{F}|)^d.
\]

We can plug this expression in Theorem 4.3 and derive the following corollary:

---

8We mention that unlike the current lower bound, the lower bounds in [12] apply to explicit functions and achieve better constants. These tools also provide fine-grained lower bounds for asymmetric protocols in which one of the two CDS parties has short communication.
Corollary 4.7 (Lower bound for $d$-arithmetic AVP). Let $Q$ be an AVP protocol whose prover is compatible with degree-$d$ polynomials over $\mathbb{F}$ for functions in $\mathcal{F}$. Denote by $A_f$ the advice size of $Q$ for a function $f$, and by $B$ and $C$ the maximal message sizes for all functions of the verifier and prover respectively. Then for most functions in $\mathcal{F}$ it holds that

$$A_f C (2B/\log|\mathbb{F}|)^d \geq \Omega(2^n).$$

Equivalently, $A_f C (2B)^d \geq \Omega(2^n \log|\mathbb{F}|)^d$, and so the lower bound improves for larger fields.

Corollary 4.7 has several interesting implications. Consider, for example, the case of an arithmetic protocol in which the communication is polynomial, then the corollary asserts that the degree must be high (at least $\Omega(n/\log n)$). Indeed, this is the case for the AVP protocol promised by Example 1.1 that achieves polynomial communication with an arithmetic prover of degree $n$.

Remark 4.8 (re-deriving lower bounds for SSS and CDS with low-degree reconstruction). The corollary also implies previous statements about SSS and CDS protocols with degree-$d$ reconstruction, studied, e.g., in [35][20]. Indeed, such schemes yields a $d$-arithmetic AVP via the reductions from Section 3. For example, since secret-sharing with total share size of $L$, leads to an AVP with $A = O(L), B = O(L)$ and $C = 1$ (by Theorem 3.1 and Observation 3.2), we get a lower bound on the total share size of $\Omega(2^{n/(d+1)})$ that matches the lower bounds of [20]. (For small values of $d$ more refined lower bounds appear in [20]).

Similarly, for (fully-decomposable) CDS protocols with degree-$d$ reconstruction we get a lower bound of $\Omega(2^{n/(d+1)})$ on the total message length for some $n$-bit predicate. For the special case of linear CDS, i.e., $d = 1$, we get that some $n$-bit predicate requires a communication of $\Omega(2^{n/2})$ for a fully-decomposable linear CDS, and, by using Remark 3.5 $\Omega(2^{n/4})$ for 2-server linear CDS where each party holds $n/2$ bits of the input. We further note that any 2k-server linear CDS trivially yields 2-server linear CDS with the same total communication and so the lower bound of $\Omega(2^{n/4})$ holds even for multi-server CDS where the number of servers is even (over an $n$-bit predicate). Similar bounds were originally proved in [16]. Matching upper bounds for linear CDS were presented in [53][21].

Next, we prove the following lower bound for PIR protocols with $d$-arithmetic servers as a corollary. In these protocols for every fixed database the PIR servers can only perform arithmetic computations of degree at most $d$ over some finite field $\mathbb{F}$. To our knowledge, this is the first lower bound for this family of PIR protocols.

Corollary 4.9 (Lower bound for PIR with arithmetic servers). Let $\alpha$ and $\beta$ denote the maximal query and response sizes in a PIR protocol with $k$ $d$-arithmetic servers for databases of size $N = 2^n$. Then the queries and responses in the protocol must satisfy

$$(\alpha + \beta + \log k) \cdot (k \beta) \cdot (2k \alpha)^d = \Omega(N).$$

Proof. Following Theorem 3.7 we convert the PIR protocol into an AVP such that for each function $f$ it holds that $A_f \leq (\alpha + \beta + \log k), B \leq k \alpha$ and $C \leq k \beta$. The bound now follows from Corollary 4.7 by noting that $|\mathbb{F}| \geq 2$. \hfill \qed

4.4 Multi-linear Provers

The second family of provers we consider are provers based on multilinear PIR servers. By Itoh [47], $(k, \ell)$-multilinear PIR servers are $k$-server PIR protocols where every query $p$ sent from the client to any of the servers is composed of $\ell$ binary vectors $q = (q_1, \ldots, q_\ell) \in \{0, 1\}^\alpha$ and where any server’s answer (for a fixed database $y$) is of length $\beta$ and is an $\ell$-multilinear function in $q$ over $\mathbb{F}_2$. For simplicity, let us focus in the case where the q_i’s are of equal length, i.e., $\alpha/\ell$ (though everything below generalizes to the more general setting). Itoh proved that in this model the total communication complexity $k(\alpha + \beta)$ is at least $\Omega_k(N^{1/(\ell+1)})$. The proof is based on an incompressibility argument. His bound is known to be tight only for the case where $\ell = 1$. Using Theorem 4.3 we get the same bound with respect to $N$ and $\ell$. Indeed, by Section 3.3 a $(k, \ell)$-multilinear PIR with parameters $\alpha$ and $\beta$ as above translates into an AVP
with advice of length $A = \log k + \alpha + \beta$ whose prover sends $k\beta$ output bits and where each output is a multilinear function over $\ell$ vectors of length $m = \alpha/\ell$ each (corresponding to the corresponding block of the verifier’s query). Since any multilinear function over $\ell$ vectors of length $m$ is determined by its behavior over $m^\ell$ inputs (e.g., a basis in each coordinate), the AVP prover is compatible with a family of functions of size at most $(k\beta)^{\alpha/\ell}$. Therefore, by Theorem 4.3 we get that

$$(\log k + \alpha + \beta) \cdot (\alpha/\ell)^\ell \log(k\beta) = \Omega(N).$$

For a constant $k$, we get that $(\alpha + \beta)^{\ell+1} = \Omega_k(N)$ and so the total communication $k(\alpha + \beta)$ is $\Omega_k(N^{1/(\ell+1)})$ matching the lower bound of [47].

### 4.5 Proof of Theorem 4.3

Let $\mathcal{G}$ be a function family, and let $Q$ be an AVP protocol with a $\mathcal{G}$-compatible prover. Let $f \in \mathcal{F}$ be a function and denote by $A_f$ the size of the advice of $Q$ when $f$ is taken as an input. We prove that $f$ can be described by $A_f \cdot \log |\mathcal{G}|$ bits, and since the description of almost all functions requires at least $\Omega(2^n)$ bits the theorem will follow.

Let $T$ be a parameter to be fixed later. We run the AVP advisor $T$ times with $f$ taken as the input with independent random strings to produce $T$ messages $a_1, \ldots, a_T \in \{0, 1\}^{A_f}$. Then, assume we continue and run the verifier and the prover $T$ times: For every $i \in [T]$ the verifier receives $a_i$, sends to the prover a message $b_i(a_i, x)$ and gets back from the prover a message $c_i(b_i, x, f)$. Then the verifier, given $(a_i, c_i, x)$, decides whether to accept or reject the input $x$. The following properties then follow from the correctness and soundness of the AVP protocol.

- If $f(x) = 1$, then there exists a function $g \in \mathcal{G}$ such that for every $i \in [T]$ if the prover sends the message $c_i = g(b_i, x, f)$ the verifier accepts $x$ with probability at least $3/4$.
- If $f(x) = 0$, then for every $i \in [T]$ and for every function $g \in \mathcal{G}$ that may be used by the prover it holds that the probability that the verifier accepts $x$ is at most $1/4$.

We set $T$ to be $T = (v \log |\mathcal{G}|)$ for a large enough constant $v$. Let $x$ be a $0$-input of $f$. By the second property, if we use the Chernoff bound and a union bound over all functions in $\mathcal{G}$ we get that the probability (over the choice of the advice) that there is at least one $g \in \mathcal{G}$ that makes the verifier accept $x$ in more than $T/2$ runs is at most $0.01$. Since $T = (v \log |\mathcal{G}|)$, we also get (with the same Chernoff and union bounds) by the first property that for $1$-inputs of $f$ the probability that all functions $g \in \mathcal{G}$ make the verifier reject $x$ in more than $T/2$ runs is at most $0.01$.

It then follows from an averaging argument that there exist advice-strings $a^*_1, \ldots, a^*_T$ such that for at least a $0.99$-fraction of the inputs of $f$ there exists a prover-function $g \in \mathcal{G}$ that makes the verifier accept $x$ in $\geq T/2$ runs if and only if $f(x) = 1$. We use this fact to describe $f$ as follows:

- Write down $a^*_1, \ldots, a^*_T$.
- Write down the set of inputs of $f$ on which the verifier does not predict correctly the value of $f(x)$.

Formally, we denote this set by $Z$ and say that an input $x \in Z$ if: (1) $f(x) = 0$ and there exists a function $g \in \mathcal{G}$ such that when the prover employs $g$ the verifier accepts $x$ in at least $T/2$ runs of the protocol; (2) $f(x) = 1$ and for every function $g \in \mathcal{G}$ employed by the prover in all $T$ runs the verifier rejects $x$ at least $T/2$ times.

A decoder will then run the AVP protocol for every advice string in $\{a^*_1, \ldots, a^*_T\}$ and every prover in $\mathcal{G}$.

Then, it will be able to determine whether $f(x) = 1$ by asserting that $x \notin Z$, and that there exists a prover-function $g \in \mathcal{G}$ for which the verifier accepts $x$ at least $T/2$ times.

The size of $Z$ is at most $0.01 \cdot 2^n$, and there are at most $\binom{2^n}{0.01 \cdot 2^n} \lesssim 2^{h(0.01) \cdot 2^n}$ different subsets of $2^n$ elements of this size (where $h(\cdot)$ is the binary entropy function). Thus, $Z$ can be encoded with

\footnote{The code of the verifier is also required in the decoding procedure, but since the verifier is independent of $f$ it can be thought of as public information, or as a part of the code of the decoder itself.}

17
\[ h(0.01) \cdot 2^n \text{ bits, and subsequently } f \text{ can be encoded with } A_f \cdot (v \log |G|) + h(0.01) \cdot 2^n \text{ bits. Therefore, we get that for most functions it holds that } A_f \cdot \log |G| = \Omega(2^n) \text{ as desired.} \]

### 5 Separating Monotone and Non-Monotone Secret Sharing

We have seen that known SSS (and DREs) lead to AVPs with exponential communication, whereas CDS protocols lead to AVPs with sub-exponential communication. This suggests that CDS-complexity may be smaller than SSS complexity. We present the first provable separation between these two notions, and along the way we show that Csirmaz’s method does not lower-bound the CDS complexity.

#### Theorem 5.1. There exists a monotone function over \(2n\) bits that can be computed by a fully-decomposable CDS protocol with total message size \(n^{1.5+o(1)}\) but requires a total share size of \(\Omega(n^2/\log n)\) from any secret-sharing scheme.

Note that the CDS complexity of a monotone function is trivially upper-bounded by its SSS complexity since any secret-sharing scheme is a CDS protocol with empty 0-messages. Recalling that CDS is essentially a “non-monotone” variant of SSS, the theorem intuitively shows that monotonicity adds an additional overhead.

In Appendix [9] we show that a stronger separation is possible if we consider partial functions. In both theorems, the function in question is a variation on the Csirmaz’s function that was used to establish the best-known secret sharing lower bound [28].

In this section, we denote by \([n]\) the set of integers in the range \([0, n-1]\). We start by defining in two steps the monotone function that will be used to separate the two models.

#### Definition 5.2 (The Csirmaz functions). For every \(n \in \mathbb{N}\), let \(k\) be the largest integer such that \(2^k \leq n\). The function \(C_n\) is parameterized by some non-increasing ordering \((y^0, \ldots, y^{2^k-1})\) of all strings of length \(k\). Here non-increasing means that

\[\text{for every } i < i', \quad \text{it holds that } y^i \leq y^{i'} .\] (2)

The function \(C_n : \{0, 1\}^{n+k} \rightarrow \{0, 1\}\), defined in [27], is the monotone function whose min-term are \(1^i \circ 0^{n-i} \circ y^j\) for \(i = 0, \ldots, 2^k - 1\), that is, the \(i\)-th min-term contains \(i\) ones concatenated with \(n-i\) zeros, concatenated by \(y^j\).

We also define a monotone function \(C'_n :\{0, 1\}^{2n} \rightarrow \{0, 1\}\), referred to as the separating Csirmaz function, as follows: Let \(k\) be the largest integer such that \(2^k \leq n\) and \(L = \lfloor n/k \rfloor\). Denote by \(y(1), \ldots, y(L)\) \(L\) strings of length \(k\) and by \(y(i)\) the \(j\)th bit of the string \(y(i)\), and define

\[C'_n(x_1, \ldots, x_{2n-k-L}, y(1), \ldots, y(L)) = C_n(x_1, \ldots, x_n, \bigvee_{\ell=1}^L y(\ell)_1, \ldots, \bigvee_{\ell=1}^L y(\ell)_k).\]

In case \(k \cdot L < n\) the bits \(x_{n+1}, \ldots, x_{2n-k-L}\) do not influence the function and therefore are ignored.

#### Theorem 5.3 (Secret sharing lower bound). The separating Csirmaz function \(C'_n\) requires total share size of \(\Omega(n^2/\log n)\) from any secret sharing scheme that realizes it.

**Proof.** Csirmaz [27] proved that in any secret-sharing scheme realizing the function \(C_n\) the shares of the last \(k\) parties must be of total size at least \(n\). This lower bound holds for any order satisfying 2. For every \(i \in \{1, \ldots, L\}\), if we fix in \(C'_n\) the \(L-1\) sets of inputs \(y(j \neq i)\) to 0\(^k\) we will be left with a residual function over \(n + k\) bits that is isomorphic to \(C_n\). Therefore by [27] in every one of the \(L\) subsets of parties corresponding to the input parts \(y(1), \ldots, y(L)\) the total share must be at least \(n\), and the total share size for \(C'_n\) must be at least \(\Omega(nL) = \Omega(n^2/\log n)\). \(\square\)

---

\[\text{An input } x \text{ is a min-term of } f \text{ if for every } x' \leq x \text{ it holds that } f(x') = 1 \text{ if and only if } x' = x.\]

\[\text{Csirmaz [23] used the same argument to prove an } \Omega(n^2/\log n) \text{ bound for the total share size for a similar function:}\]

\[C''_n(x_1, \ldots, x_{2n-k-L}, y(1), \ldots, y(L)) = \bigvee_{i=1}^L C_n(x_1, \ldots, x_n, y(i)).\]
The following lemma will be proved in Section 5.1

**Lemma 5.4.** The function $C_n$ has a fully-decomposable CDS protocol where each server communicates at most $n^{1/2+o(n)}$ bits.

We are now ready to prove Theorem 5.1. We will use the following generic transformation based on the closure properties over AND and OR gates for secret sharing. (Recall that the message complexity of a fully-decomposable CDS is $\ell$ if each server communicates at most $\ell$ bits.)

**Claim 5.5.** Suppose that the predicate $f : \{0, 1\}^{n+1} \rightarrow \{0, 1\}$ can be realized by a fully-decomposable CDS whose message complexity is $\ell$ bits. Let $g : \{0, 1\}^{n+k} \rightarrow \{0, 1\}$ be the predicate that is obtained by substituting the last input of $f$ by the disjunction of $k$ “new” input variables, i.e.,

$$g : (y_1, \ldots, y_{n+k}) \mapsto f(y_1, \ldots, y_n, \bigvee_{i=n+1}^{n+k} y_i).$$

Then, $g$ can be realized by a fully-decomposable CDS in which each server communicates $\ell$ bits. Moreover, the same holds for substitution by a conjunction.

**Proof.** For $1 \leq i \leq n+1$ let $\text{ENC}_i : \{0, 1\} \times S \times R \rightarrow \{0, 1\}^\ell$ denote the encoding function of the fully-decomposable CDS for $f$. The CDS $(\text{ENC}_i')_{1 \leq i \leq n+k}$ for $g$ is defined as follows.

Given a secret $s$, sample randomness $r \in R$ for the original CDS and compute for every $1 \leq i \leq n+1$ the pair of $\ell$-bit messages $z_{i,0} = \text{ENC}_i(0, s, r)$ and $z_{i,1} = \text{ENC}_i(1, s, r)$. In addition, we share the message $z_{n+1,0}$ into $k$ shares by sampling $\rho_1, \ldots, \rho_{k-1} \in \{0, 1\}^\ell$ and setting $\rho_k = z_{n+1,0} - (\rho_1 + \cdots + \rho_{k-1})$. The first $n$ servers in the new CDS act as before, i.e., for $1 \leq i \leq n$ the $i$th server sends $z_{i,b}$ if its input is $b$. Each of the last $k$ servers, $i \in \{n+1, \ldots, n+k\}$, sends $z_{n+1,1}$ if $y_i = 1$ and $\rho_1$, otherwise.

We analyze the protocol. Fix an input $y$ and let $x = (y_1, \ldots, y_n, \bigvee_{i=1}^k y_i)$. By definition, $g(y) = f(x)$, and by construction, the messages that are sent in the $g$-CDS form a randomized encoding of the messages that are released in the $f$-CDS under the input $x$. In particular, if $y$ satisfies $g$ then by the correctness of the RE, given the $g$-CDS we can decode the messages of the $f$-CDS and recover the secret $s$. If $y$ does not satisfy $g$ then the messages in the $g$-CDS can be simulated based on the messages of the $f$-CDS over the input $x$, and since $f(x) = 0$, the secret remains hidden. \qed

**Proof of Theorem 5.1.** By Theorem 5.3 the total share size for $C_n'$ is at least $\Omega(n^2 / \log n)$. By Lemma 5.4, the function $C_n'$ has a fully-decomposable CDS with message length of $n^{1/2+o(1)}$, and by repeatedly applying Claim 5.5 we conclude that the same holds for $C_n'$. Overall, $C_n'$ has a CDS whose total message size is $n^{1/2+o(1)}$ as desired. \qed

### 5.1 Proof of Lemma 5.4

We now prove Lemma 5.4 in three steps. As a warm-up we describe a 2-server CDS protocol for $C_n$ with messages of size $O(n^{1/2})$. Then we show how to fully-decompose Bob’s computation, and finally we will fully-decompose Alice’s computation.

**Claim 5.6.** The predicate $C_n$ admits a 2-server CDS protocol where the first server takes as inputs the first $n$ bits, the second server takes the last $k$ bits and the message sizes are $n^{1/2} + 1$.

**Proof.** Let $C_n$ be a Csirmaz function defined over the ordered strings of length $k y_0, \ldots, y_{2^k-1}$, and let $s$ be the secret bit. We describe a 2-server CDS protocol for $C_n$ where Alice holds the first $n$ input bits and Bob holds the last $k$ input bits. The idea is to locally reduce the problem to a less-than-or-equal predicate and then construct a cheap CDS for this predicate. Details follow.

\footnote{More accurately, the mapping $(y, s, r) \mapsto (y, (\text{ENC}_i((x_i, s, r)))_{1 \leq i \leq n}$ viewed as a deterministic mapping is encoded by the randomized mapping $(y, s, r, \rho) \mapsto (y, (\text{ENC}_i((y_i, s, r, \rho)))_{1 \leq i \leq n+k}$ where $\rho = (\rho_1, \ldots, \rho_{k-1})$.}
Given an input \( x \in \{0, 1\}^n \) Alice counts the number of leading ones in \( x \) and writes this number in a \( \sqrt{n} \)-basis, i.e., \( x \) is mapped to \( (a, b) \in [\sqrt{n}] \times [\sqrt{n}] \) where \( 1^{\sqrt{n}+b} \) is the longest all-1 prefix of \( x \). Given an input \( y \in \{0, 1\}^k \) Bob computes the rank of \( y \) according to the order that is defined by the Csirmaz function and writes this number in a \( \sqrt{n} \)-basis, i.e., Bob maps \( y \) to \( (a', b') \in [\sqrt{n}] \times [\sqrt{n}] \) where 
\[
y = y^{a'\sqrt{n}+b'}.
\]
Under these mappings \( C_n(x, y) = 1 \) if and only if \( a\sqrt{n} + b \geq a'\sqrt{n} + b' \). Accordingly, we use the following CDS protocol.

<table>
<thead>
<tr>
<th>A 2-server CDS for ( C_n )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Let ( s ) be a secret bit and let ( R_0, \ldots, R_{\sqrt{n}-1} ) and ( r_0, \ldots, r_{\sqrt{n}-1} ) be random bits.</td>
</tr>
<tr>
<td>1. On input ( x \in {0, 1}^n ), Alice computes the representation ( (a, b) ) that corresponds to ( x ) and sends ( r_0, \ldots, r_{\sqrt{n}-1}, (s \oplus R_a) ).</td>
</tr>
<tr>
<td>2. On input ( y \in {0, 1}^k ), Bob computes the representation ( (a', b') ) that corresponds to ( y ) and sends ( R_{a'+1}, \ldots, R_{\sqrt{n}-1}, (R_{a'} \oplus r_{b'}) ).</td>
</tr>
</tbody>
</table>

To analyze the correctness of the protocol, fix an input \( (x, y) \) for which \( a\sqrt{n} + b \geq a'\sqrt{n} + b' \), and let us distinguish between two cases. If \( a = a' \) and \( b \geq b' \), the secret can be recovered by retrieving the bits \( s \oplus R_a, r_{b'} \) from Alice’s message and the bit \( R_a \oplus r_{b'} \) from Bob’s message, and XORing these bits together. If \( a > a' \) the secret can be recovered by retrieving the bit \( s \oplus R_a \) (sent by Alice) with the bit \( R_a \) (that is sent by Bob).

To argue that the protocol is private, let us fix an input \( (x, y) \) for which \( a\sqrt{n} + b < a'\sqrt{n} + b' \), and show that the secret remains hidden. Again we have two cases. If \( a < a' \) the CDS referee receives the message \( s \oplus R_a \) from Alice, but does not learn any information about the bit \( R_a \). If \( a = a' \) and \( b < b' \), the referee receives \( s \oplus R_a \) and \( R_a \oplus r_{b'} \) but does not learn \( r_{b'} \) and thus cannot unmask \( s \). Formally, in both cases, for any fixing of \( s \), Alice sends a random string and Bob sends an independently chosen random string, and the view of the referee is independent of \( s \).

We continue by showing how to distribute the 2-server CDS protocol described above. That is, we will replace Alice with \( n \) servers and Bob with \( k \) servers, each holding a single input bit.

Simulating Bob with \( k \) servers. Recall that in the 2-server protocol Bob takes an input \( y \in [n] \), parses it to \( (a, b) \in [\sqrt{n}] \times [\sqrt{n}] \) and outputs the bits \( R_{a+1}, \ldots, R_{\sqrt{n}} \) and \( R_a \oplus r_b \). We will simulate Bob by \( k \) servers where each server holds a single bit of \( y \). First, observe that it suffices to release a randomized encoding of Bob’s original message. Specifically, we will release the vector
\[
(R_{a+1}, \ldots, R_{\sqrt{n}}, R_a \oplus u, u \oplus r_b),
\]
where \( u \) is a random bit that is given to the servers as part of their common random string. Now, the key observation is that on every input the \( k \) servers only have to release some subset of the same \( 3\sqrt{n} \) bits
\[
R_0, \ldots, R_{\sqrt{n}-1}, (R_0 \oplus u), \ldots, (R_{\sqrt{n}-1} \oplus u), (u \oplus r_0), \ldots, (u \oplus r_{\sqrt{n}-1}).
\]
For example, on input \( (7, 5) \) (equivalently, the string \( y^{7\sqrt{n}+5} \)) the servers should release \( R_0, \ldots, R_{\sqrt{n}}, R_7 \oplus u \) and \( u \oplus r_5 \). We will think of these random bits as secrets known to all servers, and disclose them in the desired way by running \( 3\sqrt{n} \) copies of a fully-decomposable CDS where each copy is invoked with its own corresponding predicate \( P : \{0, 1\}^k \rightarrow \{0, 1\} \) and independent randomness. By employing the fully-decomposable CDS protocol of [52] that has messages of size \( 2^{O(\sqrt{\log k})} \) for \( k \)-bit predicates, and by recalling that \( k \leq \log n \), we get that each of the \( k \) servers emulating Bob communicates at most
\[
3\sqrt{n} \cdot 2^{O(\sqrt{\log k})} = n^{1/2 + o(1)}
\]
bits.
Simulating Alice with \( n \) servers. We move on to distribute the role of Alice among \( n \) servers. Recall that Alice maps an input \( x \in \{0,1\}^n \) to \((a_x,b_x) \in [\sqrt{n}] \times [\sqrt{n}]\) such that the longest all-one prefix of \( x \) is of length \( a_x \sqrt{n} + b_x \), and then she reveals the message
\[
m_x = (r_0, \ldots, r_{b_x}, (s \oplus R_{a_x})).
\]
We will replace Alice with \( n \) servers, each holding a single bit of \( x \), that release the message \( m_x \) and nothing else. To release the first \( b_x + 1 \) bits of \( m_x \), we will invoke, for each \( b \in [\sqrt{n}] \), a fully-decomposable CDS for the secret \( r_b \) with the predicate \( Q_b : \{0,1\}^n \rightarrow \{0,1\} \) that maps \( x \) to 1 if \( b \leq b_x \). To release the last bit of \( m_x \), we will invoke, for every \( a \in [\sqrt{n}] \), a fully-decomposable CDS with the secret \( (s \oplus R_a) \) and predicate \( P_a : \{0,1\}^n \rightarrow \{0,1\} \) that given \( x \in \{0,1\}^n \) outputs 1 if \( a = a_x \). Overall, we call \( 2 \sqrt{n} \) protocols. We will show that in each of these protocols every server communicates \( O(1) \) bits and so the overall complexity is \( O(\sqrt{n}) \). For this, it suffices to show (Lemma A.1) that each of the above predicates can be realized by a formula in which each variable appears in a constant number of leaves (i.e., read-\( k \) formula for \( k = O(1) \)). We will prove this in the following claim.

**Claim 5.7.** For each \( a,b \in [\sqrt{n}] \), the predicates \( P_a : \{0,1\}^n \rightarrow \{0,1\} \) and \( Q_b : \{0,1\}^n \rightarrow \{0,1\} \) can be computed by a read-once and a read-twice formula, respectively.

**Proof.** Fix \( a \in [\sqrt{n}] \). Recall that \( P_a(x) = 1 \) if the number \( t \) of leading ones in \( x \) satisfies \( a \cdot \sqrt{n} \leq t < (a + 1) \sqrt{n} \). Thus, \( P_a \) is computed by the read-once formula
\[
\left( \bigvee_{k=0}^{\sqrt{n}a-1} \neg x_k \right) \land \left( \bigvee_{k=\sqrt{n}a}^{\sqrt{n}(a+1)-1} x_k \right),
\]
where \( x = (x_0, \ldots, x_{n-1}) \) and \( x_{n-1} \) is a dummy variable that is always set to 1.

We move on to the predicate \( Q_b \) for some fixed \( b \). For \( 1 \leq i \leq \sqrt{n} \), define the function \( Q_{b,i} : \{0,1\}^{i \sqrt{n}} \rightarrow \{0,1\} \) that given \( x \in \{0,1\}^{i \sqrt{n}} \) counts the number \( t \) of leading ones in \( x \) and sets its output to 1 if \( t \mod \sqrt{n} \) is at least \( b \). We will show that, for every \( i \), the function \( Q_{b,i} \) has a read-twice formula and conclude the claim by letting \( Q_b(x) := Q_{b,\sqrt{n}}(x) \).

We implement \( Q_{b,i} \) recursively. For \( i = 1 \), we compute \( Q_{b,1} \) via
\[
\bigwedge_{k=0}^{b} x_k.
\]
Indeed, \( Q_{b,1} \) simply checks if the number of leading ones is at least \( b \), i.e., if all the first \( b \) bits are ones. For \( i > 1 \), we parse the input \( x \in \{0,1\}^{i \sqrt{n}} \) as a pair \((L,R)\) where \( L \in \{0,1\}^{(i-1) \sqrt{n}} \) contains the first \( \sqrt{n} \) bits of \( x \) and \( R \in \{0,1\}^{(i-1) \sqrt{n}} \) contains the last \( (i-1) \cdot \sqrt{n} \) bits of \( x \), and then compute
\[
Q_{b,1}(L) \land \left( Q_{b,i-1}(R) \lor \bigvee_j \neg L_j \right)
\]
(3)

By induction on \( i \), we prove that (3) is a read-twice formula that computes \( Q_{b,i} \). First, note that each variable of \( L \) is being read two times (since \( Q_{b,1} \) is a read-once formula) and each of the variables of \( R \) is being read two times (by the induction hypothesis). As for correctness, observe that if (a) the number of leading ones in \( x \) is smaller than \( \sqrt{n} \) then the output of \( Q_{b,i} \) should be \( Q_{b,1}(L) \) and if (b) the number of leading ones is at least \( \sqrt{n} \) then the output should be \( Q_{b,i-1}(R) \). Indeed, in case (a) the expression \( \bigvee_j \neg L_j \) evaluates to 1, and so (3) simplifies to \( Q_{b,1}(L) \), and in case (b), it holds that \( Q_{b,1}(L) = 1 \) and \( \bigvee_j \neg L_j = 0 \), and so (3) simplifies to \( Q_{b,i-1}(R) \), as required.

This completes the proof of Lemma 5.4.
6 Separating CDS Protocols from DRE

We prove the following theorem:

**Theorem 6.1.** There exists a predicate \( f : \{0, 1\}^{2n} \to \{0, 1\} \) that can be computed by a fully-decomposable CDS protocol with total message size \( n^{1+o(1)} \) such that any DRE for \( f \) has total message length of \( \Omega(n^2/\log n) \).

To prove the theorem we define the **Multi-Index function** \( \text{MI}_n \) (that can be viewed as a non-monotone analog of the separating Csirmaz function from Section 5), and show that it has the desired DRE lower bound and CDS upper bound.

**Definition 6.2** (The Multi-Index function). Denote by \( \text{Index}_k : \{0, 1\}^{2^k} \times \{0, 1\}^k \to \{0, 1\} \) the index function that given a “database” string \( x = (x_1, \ldots, x_{2^k}) \) and an “index” string \( y = (y_1, \ldots, y_k) \) outputs the \( y \)th bit of \( x \) (e.g., when \( y \) is interpreted as an integer according to the standard binary representation). For every \( n \in \mathbb{N} \) let \( k \) be the largest integer such that \( 2^k \leq n \), let \( L = \lfloor n/k \rfloor \) and define the **Multi-Index function** \( \text{MI}_n : \{0, 1\}^{2n} \to \{0, 1\} \) as follows. The input is parsed into \( x, y_1, \ldots, y(L) \) where \( x = (x_1, \ldots, x_{2n-k \cdot L}) \) is a string of length \( 2n - k \cdot L \) and \( y(1), \ldots, y(L) \) are strings of length \( k \), and the output is

\[
\text{Index}_k(x_1, \ldots, x_{2^k}, y_1, \ldots, y(L)) = \left( \bigvee_{\ell=1}^L y(\ell)_1, \ldots, \bigvee_{\ell=1}^L y(\ell)_k \right),
\]

that is, we compute a \( k \)-bit index \( y \) by letting \( y_j = \bigvee_{\ell=1}^L y(\ell)_j \) for every \( j \in [k] \) and output the \( y \)th entry of the database \( x \).

**A DRE lower bound for multi-index.** In [13] it is shown that the total message complexity of any DRE for a function \( f \) is lower-bounded by the Nečiporuk complexity \( G(f) \) [32]. Before defining \( G(f) \) we need to set some notation.

For a function \( f : \{0, 1\}^n \to \{0, 1\} \), a set \( S \subset \{1, \ldots, n\} \) of input variables, and an assignment \( z \in \{0, 1\}^{\overline{S}} \) to the variables outside the set \( S \), we denote by \( f_{S|z} : \{0, 1\}^S \to \{0, 1\} \) the residual function that is obtained from \( f \) by fixing the variables outside \( S \) to \( z \). Let

\[
g_S(f) := \log(|\{f_{S|z} : z \in \{0, 1\}^{\overline{S}}\}|),
\]

denote the number of different functions that can be obtained from \( f \) by fixing the variables in \( \overline{S} \). Then the Nečiporuk complexity of \( f \) is

\[
G(f) := \max_V \sum_{v \in V} g_v(f).
\]

where the maximum ranges over all partitions \( V \) of \( \{1, \ldots, n\} \).

We show that \( \text{MI}_n \) has a Nečiporuk complexity of \( \Omega(n^2/\log n) \), and thus by [13] any DRE for \( \text{MI}_n \) has total message complexity of \( \Omega(n^2/\log n) \).

**Claim 6.3.** The Nečiporuk’s complexity of \( \text{MI}_n \) is \( \Omega(n^2/\log n) \).

**Proof.** Let \( n \) be an integer, \( k \) be the largest integer such that \( 2^k \leq n \) and \( L = \lfloor n/k \rfloor \). Consider the partition of the \( 2n \) inputs of \( \text{MI}_n \) to \( L + 1 \) sets \( V_0, V_1, \ldots, V_L \) where for every \( 1 \leq i \leq L \), \( V_i \) is the set of \( k \) inputs denoted by \( y(i) \), and \( V_0 \) contains the first \( 2n - k \cdot L \) inputs denoted by \( x \). Then, for every \( 1 \leq i \leq L \), if the variables of \( V_{j \neq i} \) are set to 0, each of the \( 2^{2^k} \) possible fixings of the input bits \( x_1, \ldots, x_{2^k} \) induces a different subfunction on the bits of \( V_i \). Hence, for every \( 1 \leq i \leq L \) it holds that \( g_{V_i}(\text{MI}_n) \geq 2^{2^k} \), and the Nečiporuk’s complexity of \( \text{MI}_n \) is at least

\[
\sum_{i=1}^L \log(2^{2^k}) = \Omega(n^2/\log n),
\]

and the claim follows. \( \square \)
A CDS upper bound for multi-index. Our next goal is to realize \( M_l \) by a fully-decomposable CDS whose message complexity is \( n^{o(1)} \). We begin by realizing \( \text{Index}_k \).

**Lemma 6.4.** The predicate \( \text{Index}_k \) can be realized by a fully-decomposable CDS protocol in which each server communicates \( 2^{O(\sqrt{k} \log k)} \) bits.

**Proof.** In [53] it is shown how to realize \( \text{Index}_k \) with the complexity \( \ell = 2^{O(\sqrt{k} \log k)} \) by an “almost”-decomposable CDS. That is, they construct a \((k+1)\)-server CDS for \( \text{Index}_k \) where one server \( A \) holds the database \( x \in \{0,1\}^{2^k} \) and each server \( B_i, i \in [k] \) holds a single bit of the index \( y_i \). We will show how to distribute the message that \( A \) computes among \( 2^k \) parties \( A_j, j \in [2^k] \) each holding a single bit \( x_j \) of the database \( x \). For this, we note that the message that \( A \) computes has the following form:

\[
A(x, r, w) := r + \sum_{j=1}^{n} x_j \cdot g(w, j),
\]

where \( r, w \in \mathbb{Z}_6^\ell \) are part of the protocol’s randomness, the sum is computed over \( \mathbb{Z}_6 \) and \( g : \mathbb{Z}_6^\ell \times [2^k] \rightarrow \mathbb{Z}_6^\ell \). Instead of computing \( A \) we let each server \( A_i, i \in [2^k] \), compute

\[
A_i(x_i, r, w; (v_1, \ldots, v_{2^k-1})) := v_i + x_i \cdot g(w, i),
\]

where \( v_1, \ldots, v_{2^k-1} \) are sampled uniformly at random from \( \mathbb{Z}_6^\ell \) and \( v_{2^k} = r - \sum_{i < 2^k} v_i \), and arithmetic operations are over \( \mathbb{Z}_6 \). To see that the protocol is a CDS for \( \text{Index}_k \), observe that \((A_i(x_i, r, w; (v_1, \ldots, v_{2^k-1})))_{i \in [2^k]} \), forms a randomized encoding for \( A(x, r, w) \). Consequently, if \( \text{Index}_k(x, y) = 1 \) then, by the correctness of the RE, one can recover the message of \( A \) given the messages of the \( A_i \)’s by applying the decoder of the original CDS. On the other hand, if \( \text{Index}_k(x, y) = 0 \) then, by the privacy of the RE, one can simulate the message of \((A_i)_{i \in [2^k]} \) given the messages of \( A \) and so the privacy of the new CDS follows from the privacy of the original CDS.

We move on to realize \( M_l \). Recall that \( M_l \) is obtained by replacing some of the inputs to \( \text{Index}_k \) by disjunctions over blocks of fresh input variables. Thus, we can repeatedly apply Claim 5.5 on the fully-decomposable protocol for \( \text{Index}_k \) and get a protocol for \( M_l \) with the same \( 2^{O(\sqrt{k} \log k)} \) message sizes. If we plug in \( k = \lfloor \log n \rfloor \) we get that the message sizes are \( n^{o(1)} \) as desired. This completes the proof of Theorem 6.1.

### 7 From prover-lacionic AVP to relaxed-CDS

We will show that given a lacionic AVP it is possible to construct a 2-party one-way communication CDS for the “universal function” \( U_n : \{0,1\}^{2^n} \times \{0,1\}^n \rightarrow \{0,1\} \). Recall that in such a protocol Alice holds the \( 2^n \)-bit long truth table of a function \( f : \{0,1\}^n \rightarrow \{0,1\} \), Bob holds a string \( x \in \{0,1\}^n \) and a secret \( s \in \{0,1\} \), Alice sends to Bob a short randomized message \( a \) of length \( \text{poly}(n) \) and Bob generates a short (\( \text{poly}(n) \)-long) message \( e \) that, together with \( x \) and \( f \) reveals \( s \) if and only if \( f(x) = 1 \). That is, the randomized mapping \( \text{ENC} : (f, x, s) \mapsto e \) forms a CDS (as per Definition 2.3) for the universal function. In fact, we will allow some error both in the correctness and in the privacy, and show that it can be reduced to negligible while preserving a \( \text{poly}(n) \) overhead.

**The construction.** Let \( Q = (A, V, P, V_{\text{acc}}) \) be a lacionic-prover AVP. For simplicity and wlog we assume that the verifier is deterministic. (Indeed, we can always let the advisor select the randomness for the verifier and send it as part of the advice.) We define the following CDS protocol. Given \( A \) plays the role of the advisor \( A \), samples an advice \( a = A(f; r_A) \) and sends it to Bob. Given \( x \) and \( a \) Bob plays the verifier and computes a query \( b = V(a, x) \). In addition, Bob uniformly samples an \( \text{“accepting”} \) prover’s answer \( c \in \{c : A_{\text{acc}}(a, x, c) = 1\} \). Next, Bob samples a pairwise independent

\[\text{This step can be implemented efficiently since the prover's answer is of constant size. We emphasize that lacionicity will be important even if we do not care about computational complexity.}\]
hash function \( h \) that outputs a single bit and sends \( e := (b, h, h(c) + s) \) where \( s \in \{0, 1\} \) is the secret. Let \( \text{Dec} \) be a decoder that given \( x, f \) and \( e = (b, h, d = h(c) + s) \) invokes the AVP prover and computes \( c' = \mathcal{P}(b, x, f) \), then it outputs \( s' = d - h(c') \).

**Analysis.** We assume that the AVP protocol is regular in the sense that for every \( f, x \) and \( a \) the number of accepted answers \( L_{f,x,a} = |\{ c : A_{acc}(a, x, c) = 1 \} | \) is some fixed number \( L \) that is independent of \( f, x \) and \( a \). (Indeed, all known AVP protocols satisfy this assumption.) We denote the soundness error of the protocol by \( \delta \), and note that since the prover’s answer is of constant size both \( \delta \) and \( L \) are constants. We analyze the protocol. From now on, let us fix some \( x \) and \( f \).

**Claim 7.1** (correctness). Suppose that \( f(x) = 1 \). Then for every \( s \in \{0, 1\} \) it holds that the random variable \( (b, h, d) = \text{Enc}(f, x, s) \) induced by the randomized protocol, satisfies

\[
\Pr[\text{Dec}(f, x, (b, h, d)) = s] \geq 0.5 + 0.5 \cdot L^{-1}.
\]

**Proof.** When \( c = c' \) the output is always correct, and when \( c \neq c' \) the output is correct with probability exactly half since \( h(c) \) and \( h(c') \) are uniformly and independently distributed. Therefore, the success probability is \( L^{-1} + (1 - L^{-1}) \cdot 0.5 = 0.5 + 0.5 \cdot L^{-1} \).

**Claim 7.2** (privacy). Suppose that \( f(x) = 0 \). Then the distribution \( (b, h, d|s = 0) \) induced by the randomized protocol with \( s = 0 \), is \( \beta \)-close to the distribution \( (b, h, d|s = 1) \) induced by the randomized protocol with \( s = 1 \), where \( \beta = \sqrt{2\delta/L} \).

Before proving Claim 7.2, we need the following simple claim.

**Claim 7.3.** Let \( \mathcal{L} \) be a random variable that is distributed over sets of strings where each set is of cardinality at least \( L \), and let \( Z \) be a jointly distributed random variable such that for every unbounded adversary \( P^* \) it holds that \( \Pr[P^*(Z) \in \mathcal{L}] \leq \delta \). Let \( C \) be a uniformly sampled element from \( \mathcal{L} \), then for every unbounded adversary \( P^* \) it holds that \( \Pr[P^*(Z) \in \mathcal{L}] \leq \delta/L \), equivalently,

\[
\mathbb{E}_{z \in \mathcal{L}}[\max_c \Pr[z|c] \in \mathcal{L}] \leq \delta/L.
\]

**Proof.** Fix \( P^* : \{0, 1\}^* \to \{0, 1\}^* \). We use \( \lambda \) and \( z \) to denote fixed outcomes of \( \mathcal{L} \) and \( Z \). Letting \( w_{z,\lambda} := \Pr[Z = z \wedge \mathcal{L} = \lambda] \) and \( C_\lambda \) denote the uniform distribution over the set \( \lambda \), we can write \( \Pr[P^*(Z) = C] \) as

\[
\sum_{z,\lambda : P^*(z) \in \mathcal{L}} w_{z,\lambda} \cdot \Pr[P^*(z) = C_\lambda] \leq \sum_{z,\lambda : P^*(z) \in \mathcal{L}} w_{z,\lambda} \cdot L^{-1} = \Pr[P^*(Z) \in \mathcal{L}] \cdot L^{-1} \leq \delta/L.
\]

The “equivalently” part follows by considering an optimal adversary \( P^*(z) \) whose output \( c \) maximizes \( \Pr[C = c|Z = z] \), and by noting that \( \Pr[P^*(Z) = C] = \mathbb{E}_{z \in \mathcal{L}}[\max_c \Pr[c|Z = z]] \).

**Proof of Claim 7.2** Fix some bit \( s \). Consider a random execution and let \( a, b, c, h \) be the corresponding random variables. Apply Claim 7.3 where \( b \) plays the role of the random variable \( Z \), \( c \) plays the role of the random variable \( C \), and the set \( \{ c' : A_{acc}(a, x, c') = 1 \} \) plays the role of the random variable \( \mathcal{L} \). By the soundness of the AVP, the probability of hitting an element in \( \mathcal{L} \) given the random variable \( Z \), is at most \( \delta \). Hence, by the above claim, the average min-entropy \( [29] \) of \( C \) given \( Z \), which is defined to be

\[
- \log \left( \mathbb{E}_{z \in \mathcal{L}}[\max_c \Pr[C = c|Z = z]] \right),
\]

is at least \( \log(L/\delta) \). Hence, by the generalized leftover-hashing lemma \([29]\), the random variable \( (b, h, h(c)) \) is \( \sqrt{\delta/(2L)} \)-close to \( (b, h, u) \) where \( u \) is an independent random bit. The claim follows for \( \beta = 2\sqrt{\delta/(2L)} = \sqrt{2\delta/L} \).
**Amplification.** When \( f(x) = 1 \) our CDS succeeds in decoding with probability \( 0.5 + 0.5\alpha \) for \( \alpha = 1/L \), whereas when \( f(x) = 0 \) no adversary can distinguish between an encoding of 0 to an encoding of 1 with an advantage better than \( \beta = \sqrt{2\delta/L} \). As shown in [3], based on the polarization lemmas of [57, 41], if \( \alpha^2 > \beta \) then one can reduce the privacy error and correctness error as per Definition 2.4 to \( 2^{-k} \) with a multiplicative overhead which is polynomial in \( k \) and in \( 1/(\alpha^2 - \beta) \). In our setting, \( \alpha \) and \( \beta \) are constants (since \( L, \delta \) are constants) and the condition \( \alpha^2 > \beta \) simplifies to

\[
L < (1/2\delta)^{1/3}. \tag{4}
\]

When there is a unique prover’s answer (\( L = 1 \)), this condition always holds (for any value of \( \delta \)). This is indeed the case in the AVP constructions that are based on CDS, SSS, and randomized encoding. Equation (4) also follows by assuming that \( c \), the bit-length of the prover’s answer, is sufficiently short as a function of the soundness error \( \delta \), i.e.,

\[
c < (4\log(1/\delta) - 1)/3. \tag{5}
\]

Indeed, if the soundness error is at most \( \delta \) then \( L \leq \delta 2^c \) since otherwise a randomly chosen prover’s answer will be accepted with probability more than \( \delta \). Therefore, (5) implies (4). This completes the proof of Theorem 1.4.

**Extension to non-regular protocols.** Assume that the protocol is “somewhat regular” in the sense that, for every \( f, x, a \) it holds that the number of accepted answers \( L_{f,x,a} \) is bounded in an interval \([L_{\min}, L_{\max}]\) where \( L_{\max}/L_{\min} \leq K \) for some constant \( K \). Then, the proof of Claim 7.1 can be generalized to show that correctness holds with probability at least \( 0.5 + 0.5L_{\max}^{-1} \) and Claim 7.2 implies that privacy can be guaranteed up to statistical distance of \( \beta = \sqrt{2\delta L_{\min}^{-1}} \). In this case, the condition \( \alpha^2 > \beta \) follows by slightly strengthening (4) to \( L_{\max} < (1/2K\delta)^{1/3} \) which translates to a stronger condition on the bit-length \( c \), i.e., \( c < (4\log(1/\delta) - \log(2K))/3 \). The argument can be naturally extended to the case where \( L_{f,x,a} \in [L_{\min}, L_{\max}] \) for most \( f, x \) and \( a \), at the expense of relaxing the CDS conditions to hold over most inputs.

**References**


---

\[14\] The amplification of [3] was presented for the setting of non-interactive CDS but it works in the current setting of one-way communication CDS (as well as in the more general setting of interactive CDS) as well.

\[15\] For PIR-based AVP, this transformation typically fails since \( L \) is typically exponential in the number of servers \( k \), whereas \( \delta = 1/k \).


respectively. Say that formulas and CDS protocols for both servers are of size at most \( \sigma \) and every variable and its negation appear as a leaf at most \( \sigma \). Thus the total communication complexity is \( O(2^{n/2}) \).

The advisor sends the verifier the hashed values of all blocks, i.e.,

\[ a_i = h(f_i), \quad \forall i \in [\sqrt{N}], \]

with \( a_0 \) which is set to be the description of the hash function \( h \). The prover then replies with the block \( f_i \) in which the input \( x \) lies. Then verifier outputs 1 iff \( h(f_i) = a_i \).

We omit the simple analysis and note that the description of \( h \) can be as small as \( \sqrt{N} \) (e.g., by employing Toeplitz-based hashing). Thus the total communication complexity is \( O(2^{n/2}) \).

A.2 Formulas and CDS Protocols

**Lemma A.1** (Formulas and CDS protocols). Let \( g : \{0,1\}^n \rightarrow \{0,1\} \) be a function that can be computed by a formula where negations only appear at the bottom and every variable \( x \) and its negation \( \neg x \) appear as a leaf at most \( \sigma \) and \( \overline{\sigma} \) times respectively. Then \( g \) has a fully-decomposable CDS protocol such that for every \( i \in [n] \) the 1-message and 0-message of the \( i \)th server are of size at most \( \sigma \) and \( \overline{\sigma} \) respectively.

**Proof of Lemma A.1** The proof is by induction on the size of the formula that we denote by \( t \). Let \( s \) be the CDS secret bit and denote by \( x_i \) the \( i \)th bit of the input \( x \). For \( t = 1 \), the formula is either \( x_i \) or \( \neg x_i \) for some \( i \in [n] \). Then in a CDS protocol for \( g \) the secret will be given as is to the 1-message or to the 0-message of the \( i \)th server. To prove the induction step, let \( G \) be a formula for \( g \) where negations only appear at the bottom and every variable and its negation appear as a leaf at most \( \sigma \) and \( \overline{\sigma} \) times respectively. Say that \( G \) is of size \( t \), and that it has the form \( G = G_1 \land G_2 \) or \( G = G_1 \lor G_2 \) where \( G_1 \) and \( G_2 \) are formulas of size \( t_1, t_2 < t \). For the case of an AND gate, we additively secret-share the secret \( s \) into random \( s_1 \) and \( s_2 \) subject to \( s_1 + s_2 = s \) and use a fully-decomposable CDS protocol for \( G_1 \) with the secret \( s_1 \) and for \( G_2 \) with the secret \( s_2 \) (we say that a CDS protocol realizes a formula \( G' \) if it realizes the function computed by \( G' \)). For the case of an OR gate, we use fully-decomposable CDS protocols for both \( G_1 \) and \( G_2 \) with the secret \( s \).
By the induction hypothesis, each of the functions computed by $G_1$ and $G_2$ has a fully-decomposable CDS protocol with message sizes that are at most the number of times each variable appears as a leaf in the formula. The proof then follows from the fact that the leaves of $G$ are exactly the combination of leaves of $G_1$ and $G_2$.

\[\square\]

### A.3 Randomness Sparsification for AVP Protocols

**Theorem A.2.** If there exists an AVP protocol for function in $F$ with messages of maximal sizes $A$, $B$ and $C$ and soundness error $s$ then for every $\varepsilon > 0$ there exists an AVP protocol for functions in $F$ with a deterministic verifier, messages of maximal size $O_\varepsilon(A + B)$, $B$ and $C$ and soundness error $s + \varepsilon$.

**Proof.** By Lemma A.3 stated below, for every $\varepsilon > 0$ we can decrease the size of the verifier’s randomness to $2B + \log B + \log(A + n) + 2\log(1/\varepsilon)$ while increasing the soundness error to $s + \varepsilon$. Then the verifier can be made deterministic by having the advisor draw its randomness $r_V$ in its place and send it as a part of the advice. The advice size will then be

$$A + 2B + \log B + \log(A + n) + 2\log(1/\varepsilon) = O_\varepsilon(A + B).$$

The equality holds since by Theorem 4.1 the size of the original advice $A$ must be of size at least $n - o(1)$.

\[\square\]

We prove the following lemma:

**Lemma A.3.** Suppose there exists an AVP protocol for every function $f \in F$ with verifier-randomness of length $\rho$ and soundness error $s$, where the maximal sizes of the messages of the advisor and verifier are $A$ and $B$ respectively. Then, for any $\varepsilon > 0$ there exists an AVP protocol with the same message sizes, soundness error $s + \varepsilon$ and verifier-randomness

$$2B + \log B + \log(A + n) + 2\log(1/\varepsilon).$$

This lemma is similar to the sparsification lemma for CDS that appears in [12], which in turn is based on the classical theorem of Newman for communication complexity [56]. The proof is taken almost verbatim from [12]. We denote $\{0, 1\}^\rho$ by $R$ and $\{0, 1\}^B$ by $B$ and describe the proof using non-Boolean PRGs [30].

**Definition A.4 (non-Boolean PRG).** Let $D = \{ D \}$ be a class of functions from $R$ to $B$ and let $\varepsilon > 0$ be an error parameter. We say that $G : L \rightarrow R$ is a non-Boolean PRG with error $\varepsilon$ against $D$ (in short, $(D, \varepsilon)$-nbPRG) if for every $D \in D$, $G$ $\varepsilon$-fools $D$, i.e., the statistical distance between $D(G(U_L))$ and $D(U_R)$ is at most $\varepsilon$, where $U_X$ stands for the uniform distribution over the finite set $X$.

**Claim A.5.** Let $Q = (A, V, P)$ be an AVP protocol for functions in $F$ with soundness error $s$, verifier-randomness from $R$ and verifier messages of maximal size $B$. Let $D$ be the class of functions from $R$ to $B$ obtained by restricting the computation made by the verifier for all possible advice strings $a \in \{0, 1\}^A$ and inputs $x \in \{0, 1\}^n$. Let $G : L \rightarrow R$ be a $(D, \varepsilon)$-nbPRG. Then, the AVP protocol $Q' = (A, V', P)$ where the verifier computes the function $V' : \{0, 1\}^n \times \{0, 1\}^A \times L \rightarrow B$ defined as

$$V'(x, a; r) = V(x, a; G(r))$$

has the same message sizes as $Q$ and soundness error of $s + \varepsilon$.

**Proof.** By the properties of the nb-PRG $G$, for every advice $a \in \{0, 1\}^A$ and input $x \in \{0, 1\}^n$ it holds that the distributions induced by the computations of the original and new verifier

$$V'(x, a; r) \quad \text{and} \quad V(x, a; r') \quad \text{where} \quad r \leftarrow_L \mathcal{L}, r' \leftarrow_R \mathcal{R}$$

are $\varepsilon$-close. Therefore, a malicious prover will be able to “fool” the verifier $V'$ with additional probability at most $\varepsilon$ compared to the original AVP protocol $Q$. The perfect correctness of the protocol is preserved as for every $x$ and $a$ the support of $V'(x, a; r)$ is a subset of the support of $V(x, a; r')$. \[\square\]
independent sequence

In a nutshell, the secret-sharing lower bounds still apply to the partial functions since the partial domain of the Csirmaz function $C$ is undefined.

of the database of are permuted “according to $\omega$” in the order $V$ given an input integer $U$ (and therefore $D_n$ and $D'_n$) are parameterized by some non-increasing order $\omega$ over all strings of length $k$: $(y_0, \ldots, y^{2^k-1})$. Define the function $g_\omega: [2^k] \rightarrow [2^k]$ as the function that given an input integer $U \in [2^k]$ computes its standard binary representation as a string $u \in \{0,1\}^k$, and outputs the integer $V$ such that $y^V = u$. That is, the binary string that corresponds to $U$ is the $V$’th string in the order $\omega$.

We now show that on its partial domain, $D_n$ equals a version of the index function where the entries of the database of are permutated “according to $\omega$”. As before, we denote by Index$_k: \{0,1\}^2 \times \{0,1\}^k \rightarrow \{0,1\}^{2^k}$

\footnote{In a nutshell, the secret-sharing lower bounds still apply to the partial functions since the partial domain of $D_n$ contains a long independent sequence, as defined in [24].}
the index function that given a “database” string \(x = (x_1, \ldots, x_{2k})\) and an “index” string \(y = (y_1, \ldots, y_k)\) outputs the \(y\)th bit of \(x\). We prove the following claim:

**Claim B.3** \((D_n \text{ and the Index function})\). Let \(D_n\) be a partial Csirma function defined by an order \(\omega\). Denote by \(D_i\) the \(i\)'th bit in a string \(x\), and let \(\Pi : \{0, 1\}^2 \to \{0, 1\}^2\) be the permutation that takes as input a database \(x\) and outputs a database \(\Pi(x)\) where for every index \(j \in [2^k]\) it holds that \(\Pi(x)_j = x_{g_\omega(j)}\). Then, for every input in the domain of \(D_n\) it holds that \(D_n(x, y) = \text{Index}_k(\Pi(x), y)\).

**Proof.** Let \((x, y) \in \{0, 1\}^{2^k} \times \{0, 1\}^k\) be the first \(2^k\) and last \(k\) bits of an input to \(D_n\), and let \(Y \in [2^k]\) be the integer that corresponds to \(y\) by the standard binary encoding. If \(D_n(x, y) = 1\) then by definition it holds that \(x = 1^j \circ 0^{n-j}\) and \(j \geq g_\omega(Y)\). Therefore,

\[
\text{Index}_k(\Pi(x), y) = \Pi(x)_Y = x_{g_\omega(Y)} = 1.
\]

If \(D_n(x, y) = 0\) then by definition it holds that \(x = 1^j \circ 0^{n-j}\) and \(j < g_\omega(Y)\). Therefore,

\[
\text{Index}_k(\Pi(x), y) = \Pi(x)_Y = x_{g_\omega(Y)} = 0
\]
as desired. \(\Box\)

By Claim \(\text{B.3}\) it suffices to realize \(\text{Index}_k(\Pi(x), y)\) with a fully-decomposable CDS protocol in order to realize \(D_n\). Since \(\Pi\) merely permutes the entries of \(x\) according to \(g_\omega\), it is possible to realize \(\text{Index}_k(\Pi(x), y)\) by having each server \(i \in [2^k]\) simulate the role of the \(g^{-1}_\omega(i)\)'th server in a fully-decomposable CDS protocol for \(\text{Index}_k(x, y)\). Recall from the proof of Theorem 6.1 that \(\text{Index}_k(x, y)\) has a fully-decomposable CDS protocol with message size \(2^{O(\sqrt{k} \log k)}\), and so we get a fully-decomposable CDS protocol for \(D_n\) with the same message size.

We move on to realize \(D'_n\). Similarly to the multi-index function in Section 6, \(D'_n\) is obtained by replacing some of the inputs to \(D_n\) by disjunctions over blocks of fresh input variables. Thus, we can repeatedly apply Claim \(\text{B.3}\) on the fully-decomposable protocol for \(D_n\) and get a protocol for \(D'_n\) with the same \(2^{O(\sqrt{k} \log k)}\) message sizes. If we plug in \(k = \lceil \log n \rceil\) we get that the message sizes are \(n^{o(1)}\) as desired. This completes the proof of Theorem \(\text{B.1}\). \(\Box\)