A preliminary version of this paper appears in Proceedings *of the* 35th *Symposium on Foundations of Computer Science*, IEEE, 1994.

Randomness-Efficient Oblivious Sampling

Mihir Bellare[*] John Rompel[†]

November 1994

Abstract

We introduce a natural notion of *obliviousness* of a sampling procedure, and construct a randomness-efficient oblivious sampler. Our sampler uses $O(l + \log \delta^{-1} \cdot \log l)$ coins to output $m = \text{poly}(\epsilon^{-1}, \log \delta^{-1}, \log l)$ sample points $x_1, \ldots, x_m \in \{0,1\}^l$ such that

$$\Pr\left[\, \left| \tfrac{1}{m} \textstyle\sum_{i=1}^{m} f(x_i) - \mathbf{E}[f] \right| < \epsilon \,\right] \;\geq\; 1 - \delta$$

for any function $f\colon\ \{0,1\}^l \to [0,1]$.

We apply this sampling procedure to reduce the randomness required to halve the number of rounds of interaction in an Arthur Merlin proof system. Given a $2g(n)$ round AM proof for L in which Arthur sends $l(n)$ coins per round and Merlin responds with a $q(n)$ bit string, we construct a $g(n)$ round AM proof for L in which Arthur sends $O(l + (q + \log g) \cdot \log l)$ coins per round and Merlin responds with a poly(n) bit string.

[*]Department of Computer Science & Engineering, Mail Code 0114, University of California at San Diego, 9500 Gilman Drive, La Jolla, CA 92093. E-mail: `mihir@cs.ucsd.edu`

[†]Work done while the author was at MIT. Present whereabouts unknown.

1 Introduction

We study sampling methods from a new angle. We introduce a natural notion of "obliviousness" of a sampling method, defined formally below, and ask what is the complexity of oblivious sampling.

The aspect of the complexity that interests us most is the number of random bits used. Accordingly, we construct a new sampling method which has the obliviousness property while being randomness-efficient.

Our motivation came from interactive proofs. We were interested in questions like: how much extra randomness do we need in order to reduce the number of rounds of interaction in an interactive proof by a constant factor? Oblivious sampling seemed crucial, and, in particular, we apply our new sampling procedure to prove a randomness-efficient version of the well-known "speedup," or round-halving theorem of Babai and Moran [4].

We hope this work will have further applications in this area, particularly for multi-prover or probabilistically checkable proofs.

There are two side aspects of this work which we feel might be interesting or useful. The first is a simpler and more direct proof of the speedup theorem of [4], obtained in the process of reducing the randomness used. The second is a pair of tail inequalities for low independence random variables which are simple both to state and to prove.

1.1 Randomness-efficient sampling

A security parameter n is ubiquitous in the following discussion. Parameters l, m are integer valued functions of n, and ϵ, δ are $[0,1]$ valued functions of n, such that $l(n), \epsilon^{-1}(n), \log \delta^{-1}(n)$ are at most $\text{poly}(n)$.

Approximation, obliviousness and our result

We are interested in procedures which enable us to approximate the average value

$$\mathbf{E}[f] \ \stackrel{\text{def}}{=}\ 2^{-l(n)} \cdot \textstyle\sum_{x\in\{0,1\}^{l(n)}} f(x)$$

of an arbitrary function $f\colon \{0,1\}^{l(n)} \to [0,1]$. Typically, such a procedure has two stages; it is accordingly specified by a pair $A = (S, E)$ of polynomial time algorithms. The first algorithm, called the *sampler*, is probabilistic, and the second, called the *estimator*, is assumed here, for simplicity, to be deterministic. To estimate $\mathbf{E}[f]$ using $A = (S, E)$, we proceed as follows. First, we run the sampler S on input 1^n to produce a sequence $x_1, \ldots, x_{m(n)} \in \{0,1\}^{l(n)}$ of *sample points*. Then we compute $y_i = f(x_i)$ for $i = 1, \ldots, m(n)$, and run E on input $y_1, \ldots, y_{m(n)}$ to get an *estimate* $\text{Est}_f^A \in \mathsf{R}$.

We say that $A = (S, E)$ is a *universal* (l, ϵ, δ)*-approximator* if for all n and all $f\colon \{0,1\}^{l(n)} \to [0,1]$ it is the case that

$$\Pr\left[\,\left|\text{Est}_f^A - \mathbf{E}[f]\right| < \epsilon(n)\,\right] \ \geq\ 1 - \delta(n)\ , \tag{1}$$

the probability being over the coin tosses of the sampler. (The universality refers to the fact that the approximator works for all functions, not some subset like just the boolean functions).

The so-called *standard* procedure is the paradigm. Here the sampler simply outputs uniformly and independently distributed sample points, and the estimator, defined by

$$E_{\text{st}}(y_1, \ldots, y_{m(n)}) \ =\ \tfrac{1}{m(n)} \textstyle\sum_{i=1}^{m(n)} y_i\ ,$$

outputs the average of the function values on the sample points as the estimate. A simple application of a Chernoff bound, such as that of Lemma A.3, shows that if $m = \Omega(\epsilon^{-2} \log \delta^{-1})$ then (1) indeed holds, for any $f\colon \{0,1\}^{l(n)} \to [0,1]$.

The particular simple form of estimation used in the standard procedure seems natural, and is crucial to some applications such as those in this paper. Accordingly, we say that S is a *universal* (l, ϵ, δ)*-oblivious sampler* if $A = (S, E_{\mathrm{st}})$ is a universal (l, ϵ, δ)-approximator. That is, it suffices to use as estimate the average of the function values on the sample points.

The drawback of the standard sampling procedure is its complexity, in particular the amount of randomness used: this is $lm = O(l\epsilon^{-2} \log \delta^{-1})$ coin tosses, too much for some applications. Thus much research has been invested in devising randomness-efficient samplers. The conclusion from the prior work, reviewed below, is that saving randomness seems finally to have been at odds with obliviousness in that the most randomness-efficient known sampler, namely that of [5], is non-oblivious. The following theorem addresses this gap.

Theorem 1.1 Let $l\colon \mathsf{N} \to \mathsf{N}$ and $\epsilon, \delta\colon \mathsf{N} \to (0,1]$. Then there exists a universal (l, ϵ, δ)-oblivious sampler which uses only $O(l + \log \delta^{-1} \cdot \log l)$ coin tosses and outputs $\mathrm{poly}(\epsilon^{-1}, \log \delta^{-1}, \log l)$ sample points.

We stress that the estimation error can be made polynomially small, namely $\epsilon(n) = 1/\mathrm{poly}(n)$, and the error probability can be made *exponentially* small, namely $\delta(n) = 2^{-\mathrm{poly}(n)}$, while maintaining $\mathrm{poly}(n)$ coins and sample points.

An open question is whether there exists a universal (l, ϵ, δ)-oblivious sampler which uses only $O(l + \log \delta^{-1})$ coin tosses and outputs $\mathrm{poly}(\epsilon^{-1}, \log \delta^{-1}, l)$ sample points. We believe the answer is affirmative. Another problem is to reduce the number of sample points, ideally to $O(\epsilon^{-2} \log \delta^{-1})$. Note that this number of sample points would be optimal, and, for constructions using this number of sample points, $O(l + \log \delta^{-1})$ coin tosses is optimal, both up to constant factors [9].

Prior work in approximation

A comparison of universal approximators based on their complexity involves consideration of many parameters. We will stress whether or not the sampler is oblivious, the number of coins used, and the number $m(n)$ of sample points used. We prefer methods where it is possible to achieve estimation error $\epsilon(n) = 1/\mathrm{poly}(n)$ and error probability $\delta(n) = 2^{-\mathrm{poly}(n)}$ with $\mathrm{poly}(n)$ coins and sample points, which means the number of sample points should be $\mathrm{poly}(\epsilon^{-1}, \log \delta^{-1}, l)$. The known results are discussed below and summarized in Figure 1, with ours in the last line for comparison.

First, as already indicated, the sampler of the standard method is oblivious, but uses too many coins.

Generating the sample points pairwise independently [11] yields a large savings in randomness and preserves obliviousness, but the number of sample points grows to $\mathrm{poly}(\epsilon^{-1}, \delta^{-1}, l)$. (Recall, we wanted $\mathrm{poly}(\epsilon^{-1}, \log \delta^{-1}, l)$ sample points).

Generate the $m(n)$ sample points by a $m(n)$ step random walk on a constant degree expander graph with vertex set $\{0,1\}^{l(n)}$; an application of Gillman's Chernoff bound for expanders [13] shows that this yields a universal oblivious sampler for $m = \Omega(\epsilon^{-2} \log \delta^{-1})$.[1] But now the randomness has grown, to $l + O(m) = O(l + \epsilon^{-2} \log \delta^{-1})$.

The approximation method of [5] uses both pairwise independence and expanders. The sampler expends only $2l + O(\log \delta^{-1})$ coins to generate $m = O(\epsilon^{-2} \log \delta^{-1})$ sample points, which is optimal

[1] Gillman's analysis improves those of [2, 12, 18]. The latter works showed that random walks on expanders have some useful sampling properties, but didn't show them capable of approximating the average value of a function in the sense of (l, ϵ, δ)-approximation we are considering here.

Due to	Method	Oblivious?	Coins	Sample points (m)
Standard	Full independence	Yes	$O(l\,\epsilon^{-2}\log\delta^{-1})$	$m_{\mathrm{st}} = O(e^{-2}\log\delta^{-1})$
[11]	Pairwise Independence	Yes	$O(l)$	$O(\epsilon^{-2}\delta^{-1})$
[13]	Expanders	Yes	$O(l+\epsilon^{-2}\log\delta^{-1})$	m_{st}
[5]	Pair. Ind. + Expanders	**No**	$O(l+\log\delta^{-1})$	m_{st}
This paper	Iterated sampling	**Yes**	$O(l+\log\delta^{-1}\cdot\log l)$	$\mathrm{poly}(\epsilon^{-1},\log\delta^{-1},\log l)$

Figure 1: **Sampling techniques.** We recover the obliviousness lost by [5] while using only an extra logarithmic factor in coin tosses.

up to constant factors [9]. But this sampler is NOT oblivious. The estimate is not the average of the function values on the sample points; rather, it is computed as follows. The estimator views its inputs $y_1,\ldots,y_{m(n)}$ as grouped into $v = O(\log\delta^{-1})$ blocks, each with $t = O(\epsilon^{-2})$ points. For each block $j = 1,\ldots,v$, it computes the average value of the points in this block, namely

$$\gamma_j \;=\; \tfrac{1}{t}\textstyle\sum_{i=(j-1)t+1}^{jt} y_i \;.$$

It then outputs the median of the values $\gamma_1,\ldots,\gamma_v$. Thus the reduction in randomness was at the cost of obliviousness.

Goldreich and Wigderson [14] reduce the randomness from $2l + O(\log\delta^{-1})$ to $l + O(\log\delta^{-1})$, but they use, in their final step, the method of [5], so that their sampler too is non-oblivious.

Zuckerman [25] has pointed out that an oblivious sampler can be derived from the space bounded generator of Nisan [20].[2] The complexity is quite good, but the construction only works when the given parameters l,ϵ,δ are restricted to have certain relations to each other. Specifically, assume $l = \Theta(m_{\mathrm{st}}/\epsilon)$, where $m_{\mathrm{st}} = \Theta(\epsilon^{-2}\log\delta^{-1})$ is the number of sample points of the standard method. Then the construction uses $O(\log(m_{\mathrm{st}})\cdot(l+\log\delta^{-1}))$ coins and m_{st} sample points. The idea is to define an FSM (cf. [20, Section 2.1]) which has space and block size $O(\log(m_{\mathrm{st}}/\epsilon)+l+\log\delta^{-1})$ and which keeps track of the average value of f on the input sequence to within a precision of ϵ/m_{st}. One then applies [20, Lemma 3].

1.2 Randomness-efficient speedup

An Arthur-Merlin game [3] is a two-party protocol. The players are an all-powerful, but, without loss of generality, deterministic, "prover," called *Merlin*, and a probabilistic, polynomial-time "verifier," called *Arthur*. The game is played on a common input, denoted w, and the purpose of the game is to convince Arthur that w belongs to some pre-determined language L. There is a designated start player, after which the planers alternate moves, each, at his turn, sending a message to the other. Merlin computes his message as a function of all previous messages, but Arthur's role is restricted: when it is his turn to play, he just tosses some coins, randomly and independently of all previous coins, and sends their outcome to Merlin. At the end of the game, he evaluates a polynomial-time predicate, applied to the common input and the full transcript of the interaction,

[2] Later constructions of space bounded generators [21] don't seem to help to do better.

in order to decide whether to accept or reject.

An Arthur-Merlin (AM) game is said to be an AM proof system for the language L if the error probability on any input w (the probability that Arthur accepts if $w \notin L$ or rejects if $w \in L$) is at most $1/3$. AM denotes the class of languages possessing Arthur-Merlin proof systems. Clearly, Arthur-Merlin games are special kinds of interactive proof systems [15]. Their special properties make them very useful for proving structural results [1, 4, 8], and on the other hand we lose none of the language recognition power: $\mathsf{AM} = \mathsf{IP} = \mathsf{PSPACE}$ [16, 19, 24].

Arthur's message length is the number of (random) bits he sends in each of his moves, and we usually denote it by $l = l(n)$, where $n = |w|$ is the length of the common input. Merlin's message length is the number of bits he sends in each of his moves, and we usually denote it by $q = q(n)$. For simplicity this discussion focuses on games consisting of $\mathrm{poly}(n)$ *rounds*, each round consisting of an Arthur move followed by a Merlin move.

Research into the effect of varying the amount of interaction or randomness in an Arthur Merlin game arose out of the desire to understand, more quantitatively, the power of these resources as demonstrated by the $\mathsf{AM} = \mathsf{PSPACE}$ result. Following many results on interaction [1, 3, 4, 8], and the recent initiation of work on randomness [5], we would like a more unified treatment enabling us to understand tradeoffs; for example: if you want to reduce the number of rounds, how much do you pay in coins? The so-called "speedup" problem, which we now discuss, captures this question in a crisp way.

Suppose we are given a $2g(n)$ round Arthur Merlin proof system for a language L in which Arthur's message length is $l(n)$ and Merlin's message length is $q(n)$ (here g, l, q are arbitrary polynomials in n). The speedup problem is to construct a $g(n)$ round Arthur-Merlin proof system for L. Babai and Moran [4] showed that speedup is possible, but quite costly: in their construction, the length of Arthur's messages (number of random bits he sends per round) in the final $g(n)$ round game is $O(lqg^3 \log g)$, considerably more than the $l(n)$ bits it was in the original game.

Our goal is to reduce this number of random bits. We note that we are not concerned with the length of Merlin's messages to Arthur in the final $g(n)$ round game, other than that it remains polynomial; our goal is to find the minimal amount of *randomness* that will suffice to solve the basic speedup problem. Our result is the following.

Theorem 1.2 Suppose we are given a $2g(n)$ round AM proof system for L, in each round of which Arthur sends $l(n) \geq \log n$ random bits and Merlin responds with a $q(n)$ bit string. Then we can construct a $g(n)$ round AM proof system for L in each round of which Arthur sends only $O(l + (q + \log g) \cdot \log l)$ random bits and Merlin responds with a $\mathrm{poly}(n)$ bit string.

A special class of games often studied in the literature is those in which the message lengths of both parties is the same. If the starting game is of this class, then our theorem implies we can cut the number of rounds by one-half in such a way that there is only a *logarithmic* factor of increase in the number of coins flipped, while Merlin's messages remain of polynomial length: specifically, Arthur uses $O(l \cdot \log(gl))$ coins per round in the final $g(n)$ round game.

Underlying this result is a new analysis of the basic speedup procedure of [4]. This analysis is actually simpler and more direct than the original one, yielding a simpler proof of the speedup theorem. It is also tighter, showing that Arthur can use less resources in the final game. The savings in randomness is a consequence both of this better analysis and the use of Theorem 1.1.

We believe it should be possible to improve Theorem 1.2 to show that Arthur need use only $O(l + q)$ coins per round in the final game. One way to do this would be to improve the randomness complexity of the sampler of Theorem 1.1 to $O(l + \log \delta^{-1})$ and then apply Theorem 5.3.

2 Low Independence Tail Inequalities

We begin by providing some tools to analyze sums of t-wise independent random variables that will be used in the next section. Since the bounds may be useful elsewhere our discussion is general.

Low independence is a central tool in computer science. Typically we are looking at a collection of t-wise independent random variables. For simplicity let them take values between 0 and 1.

Definition 2.1 Let $X_1, \ldots, X_n$ be random variables taking values in the interval $[0,1]$. We say that $X_1, \ldots, X_n$ are t-wise independent if for any $a_1, \ldots, a_t \in [0,1]$, and any distinct indices $1 \leq i_1, \ldots, i_t \leq n$, it is the case that

$$\Pr\left[\, X_{i_1} = a_1, \ldots, X_{i_t} = a_t \,\right] \;=\; \prod_{j=1}^{t} \Pr\left[\, X_{i_j} = a_j \,\right] \,.$$

We want to bound the deviation from the mean of the sum $X = X_1 + \cdots + X_n$. For $t = 2$ a bound that is usually satisfactory is easily obtained, from Chebyshev's inequality, as follows. Let $\mu = \mathbf{E}[X]$ and use the fact that $\mathbf{Var}[X] = \sum_{i=1}^{n} \mathbf{Var}[X_i] \leq \sum_{i=1}^{n} \mathbf{E}[X_i] \leq \mu$. Then

$$\Pr\left[\, |X - \mu| \geq A \,\right] \;\leq\; \frac{\mu}{A^2} \;\leq\; \frac{n}{A^2} \,.$$

The following tail inequalities provide similar bounds for higher values of t. The first is the simplest, expressing the bound only in terms of the number n of random variables, the independence t, and the deviation A. Think of C_t as a constant: indeed it is always at most 1.0004, but actually decays exponentially in t.

Lemma 2.2 (First t-wise Independence Tail Inequality) Let $t \geq 4$ be an even integer. Suppose $X_1, \ldots, X_n$ are t-wise independent random variables taking values in $[0,1]$. Let $X = X_1 + \cdots + X_n$ and $\mu = \mathbf{E}[X]$, and let $A > 0$. Then

$$\Pr\left[\, |X - \mu| \geq A \,\right] \;\leq\; C_t \cdot \left(\frac{nt}{A^2}\right)^{t/2} \,,$$

where $C_t = 2\sqrt{\pi t} \cdot e^{1/6t} \cdot e^{-t/2} \leq 1.0004$.

We remark that $C_t \leq 1$ for $t \geq 6$, which simplifies the bound in this range.

When the expectation of X is small, the following more general tail inequality may be more useful: the bound is in terms of n, t and the expectation μ.

Lemma 2.3 (Second t-wise Independence Tail Inequality) Let $t \geq 4$ be an even integer. Suppose $X_1, \ldots, X_n$ are t-wise independent random variables taking values in $[0,1]$. Let $X = X_1 + \cdots + X_n$ and $\mu = \mathbf{E}[X]$, and let $A > 0$. Then

$$\Pr\left[\, |X - \mu| \geq A \,\right] \;\leq\; C_t \cdot \left(\frac{t\mu + t^2}{A^2}\right)^{t/2} \,,$$

where $C_t = 2\sqrt{\pi t} \cdot e^{1/6t} \cdot (5/2e)^{t/2} \leq 8$.

The basic paradigm for proving such inequalities is standard. Nonetheless our proof of Lemma 2.2, appearing in Appendix A, is interesting for its simplicity, achieved primarily by the use of integrals to express and compute the higher central moments. Lemma 2.3 is more interesting; to prove it, we had to first derive a Chernoff bound tailored to our ends (cf. Lemma A.5).

The special case of Lemma 2.2 in which $\Pr[X_i = 0] = \Pr[X_i = 1] = 1/2$ for all $i = 1, \ldots, n$ appeared in [7]. We are not aware of any other explicitly stated t-wise independent tail inequalities prior to the appearance of the above lemmas in the preliminary versions of this work [6, 22]. Since then, however, other such inequalities have appeared [23].

In what follows we will need only the first of the above lemmas.

3 Definition

The formal definition of a universal (l, ϵ, δ)-oblivious sampler that follows is somewhat more general than what was discussed in Section 1.1 in that we consider approximating a collection of functions $f_1, \ldots, f_{m(n)}$ rather than just a single function f.

Definition 3.1 Let $l, m\colon \mathsf{N} \to \mathsf{N}$ and $\epsilon, \delta\colon \mathsf{N} \to [0,1]$. A universal (l, ϵ, δ)-oblivious sampler is a probabilistic, polynomial time algorithm S which on input 1^n outputs a sequence of points $x_1, \ldots, x_{m(n)} \in \{0,1\}^{l(n)}$ such that: for any collection of $m(n)$ functions $f_1, \ldots, f_{m(n)}\colon \{0,1\}^{l(n)} \to [0,1]$ it is the case that

$$\Pr\left[\, \left| \tfrac{1}{m(n)} \sum_{i=1}^{m(n)} (f_i(x_i) - \mathbf{E}[f_i]) \right| < \epsilon(n) \,\right] \;\;\geq\;\; 1 - \delta(n) \;,$$

the probability being over the coin tosses of S.

The points $x_1, \ldots, x_m$ are called the *sample points*, and we refer to the sequence of coin tosses used by the sampler as the *seed*.

4 Construction

In this section we prove Theorem 1.1. We begin by describing t-universal hash functions, which we will use to implement the sampler. As an illustration of our methods, we first construct a simple oblivious sampler which nonetheless gives a non-trivial savings in coin tosses. We then present an iterated sampling technique which significantly reduces the number of random bits used. Finally we specify the oblivious sampler that results.

The security parameter n will usually be understood in what follows, and we write l, m, ϵ, δ instead of $l(n), m(n), \epsilon(n), \delta(n)$, respectively.

4.1 Universal hash functions

Definition 4.1 A collection H of functions mapping n bits to m bits is *t-universal* if for all distinct $x_1, \ldots, x_t \in \{0,1\}^n$ and all $y_1, \ldots, y_t \in \{0,1\}^m$, picking h at random from H implies that $(h(x_1), \ldots, h(x_t)) = (y_1, \ldots, y_t)$ with probability exactly 2^{-tm}.

For the rest of this section, $H_t(n, m)$ will denote a t-universal collection of hash functions mapping n bits to m bits in which the description of a function uses $t \cdot \max(n, m)$ bits (cf. [10]).

4.2 A Simple Oblivious Sampler

Using t-universal hash functions we can construct a very simple (l, ϵ, δ)-oblivious sampler as follows. The sampler takes as input a randomly selected element h from $H_t(d, l)$, where $d \geq \lg m$, and outputs $h(1), \ldots, h(m)$ (we are identifying $\{0,1\}^d$ with $\{1, 2, \ldots, 2^d\}$). We use the first t-wise independence tail inequality to specify m and t.

Let $Y_i = f_i(h(i))$ for $1 \leq i \leq m$. It follows from the definition of t-universal hash functions that $Y_1, \ldots, Y_m$ is a collection of t-wise independent random variables in the range $[0,1]$. Assuming for simplicity that t is an even integer ≥ 6, we apply the first t-wise independence tail inequality to get

$$\Pr[\,|Y - \mathbf{E}[Y]| \geq \epsilon m\,] \;\leq\; \left(\frac{mt}{(\epsilon m)^2}\right)^{t/2} \;=\; \left(\frac{t}{\epsilon^2 m}\right)^{t/2} .$$

But the left hand side is just

$$\Pr[\,|\textstyle\sum_{i=1}^{m} f_i(x_i) - \sum_{i=1}^{m}\mathbf{E}[f_i]\,| \geq \epsilon m\,] \;=\; \Pr\left[\,\left|\,\frac{1}{m}\sum_{i=1}^{m}(\,f_i(x_i) - \mathbf{E}[f_i]\,)\right| \geq \epsilon\,\right] .$$

Since we want this probability to be less than δ, it suffices to have

$$\delta \;\geq\; \left(\frac{t}{\epsilon^2 m}\right)^{t/2}$$

or equivalently,

$$m \;\geq\; \frac{t}{\epsilon^2 \delta^{2/t}} .$$

Restating the above, we have the following lemma.

Lemma 4.2 Let t, m, d be integers such that $t \geq 6$ is even, $m \geq \frac{t}{\epsilon^2\delta^{2/t}}$, and $d \geq \lg m$. Then for any collection of m functions $f_1, \ldots, f_m$: $\{0,1\}^l \to [0,1]$, picking h at random from $H_t(d,l)$ implies that

$$\Pr\left[\,\left|\,\frac{1}{m}\sum_{i=1}^{m}(\,f_i(h(i)) - \mathbf{E}[f_i]\,)\right| < \epsilon\,\right] \;\geq\; 1 - \delta .$$

Next, observe that this sampler uses $t \cdot \max(d,l)$ random bits. This leads us to think that we just make t as small as possible and m as large as necessary to minimize the number of bits. However, we have another constraint: m must be bounded by a polynomial in the security parameter n. Requiring that m be polynomial in n and optimizing, we get $t = \log\delta^{-1}/O(\log n)$ and thus use

$$O\left(\frac{\log\delta^{-1}}{\log n}\cdot l + \log\delta^{-1}\right)$$

random bits.

4.3 Iterated Sampling

In the previous section, we showed how to sample a collection of functions using t-universal hash functions. The number of bits we used to (l,ϵ,δ)-sample, for fixed ϵ and δ, was proportional to the logarithm of the domain size of the functions and roughly inversely proportional to the logarithm of the size of the sample. Given a set of functions with a fixed domain size, this suggested that we simply make our sample as large as is tolerable (i.e. polynomial).

In this section we will improve our bounds by iterating the sampling primitive of the previous section in a novel manner. Roughly, the idea is to take a large sample and then take a smaller sample of the first sample. Each of these samples will require many fewer bits than our original method: the first because the sample is larger; the second because we are sampling a smaller space. Our sampler becomes the composition of two randomly chosen hash functions. Note that our first sample can be superpolynomial in size—we only sample a polynomial number of its points. This idea can then be improved by taking a sequence of smaller and smaller samples instead of just two.

First we need a variant of Lemma 4.2. The reason is that we actually think of sampling each function individually except for the final sample.

Lemma 4.3 Let t, d be integers such that $t \geq 6$ is even and $2^d \geq \frac{t}{\epsilon^2 \delta^{2/t}}$. Then for any function $f\colon \{0,1\}^l \to [0,1]$, picking h at random from $H_t(d,l)$ implies that

$$\Pr[\,|\mathbf{E}[f \circ h] - \mathbf{E}[f]\,| < \epsilon\,] \;\geq\; 1 - \delta\;.$$

Proof: Let $m = 2^d$. The probability of failure is

$$\Pr[\,|\mathbf{E}[f \circ h] - \mathbf{E}[f]\,| \geq \epsilon\,] \;=\; \Pr[\,|\textstyle\sum_{i=1}^m f(h(i)) \;-m\,\mathbf{E}[f]\,| \geq \epsilon m\,]\;.$$

By the first t-wise independence tail inequality this is bounded above by

$$\left(\frac{mt}{(\epsilon m)^2}\right)^{t/2} \;=\; \left(\frac{t}{\epsilon^2 m}\right)^{t/2} \;\leq\; (\delta^{2/t})^{t/2} \;=\; \delta\;,$$

as desired. ■

Now we can combine Lemmas 4.2 and 4.3 to obtain a new sampling lemma. Our sampler will be the composition of a sequence of length doubling hash functions. This represents a sequence of samples in which the size of each sample is the square root of the size of the preceding sample.

Lemma 4.4 Let r, m, d be integers and $t_1, \ldots, t_r$ even integers ≥ 6. Suppose $d \geq \lg m$,

$$2^{2^{r-j}d} \;\geq\; \frac{t_j}{\epsilon_*^2 \delta_*^{2/t_j}} \quad (j = 1, \ldots, r-1) \quad \text{and}$$

$$m \;\geq\; \frac{t_r}{\epsilon_*^2 \delta_*^{2/t_r}}\;.$$

Then for any collection of m functions $f_1, \ldots, f_m\colon \{0,1\}^{2^r d} \to [0,1]$, picking h_j at random from $H_{t_j}(2^{r-j}d, 2^{r-j+1}d)$ implies that

$$\Pr\left[\,\left|\,\tfrac{1}{m}\textstyle\sum_{i=1}^m(\,(f_i \circ h_1 \circ \cdots \circ h_r)(i) - \mathbf{E}[f_i]\,)\,\right| < r\epsilon_*\,\right] \;\geq\; 1 - r\delta_*\;.$$

Proof: Let $f = \frac{1}{m}\sum_{i=1}^m f_i$. We first claim by induction that for $0 \leq j \leq r-1$,

$$\Pr[\,|\mathbf{E}[(f \circ h_1 \circ \cdots \circ h_j)] - \mathbf{E}[f]\,| < j\epsilon_*\,] \;\geq\; 1 - j\delta_*\;.$$

The base case ($j = 0$) is immediate, and the induction step is just Lemma 4.3, using $f \circ h_1 \circ \cdots \circ h_{j-1}$ as the function. The final step is to apply Lemma 4.2, using $\{f_i \circ h_1 \circ \cdots \circ h_{r-1}\}_{i=1}^m$ as the collection of functions. ■

We now optimize the parameters to get our final sampler (this is a more precise version of Theorem 1.1).

Theorem 4.5 Let $l\colon \mathsf{N} \to \mathsf{N}$ and $\epsilon, \delta\colon \mathsf{N} \to (0,1]$ be such that $l(n), \epsilon^{-1}(n), \log\delta^{-1}(n) \leq \mathrm{poly}(n)$. Then we can construct a universal (l, ϵ, δ)-oblivious sampler which outputs $m = O(\epsilon^{-6}\log^6 l + \log^3\delta^{-1})$ sample points using $O(l + \log\delta^{-1} \cdot \log l)$ coin tosses.

Proof: Let $m = \max(\epsilon^{-6}\lg^6 l, [12(1+\lg\delta^{-1})]^3)$. If $l \leq \lg m$ then the sampler can simply output all the points in $\{0,1\}^l$ without expending any coin tosses, so assume $l > \lg m$. Let $r = \lg(l/\lg m)$ and let

$$t_j \;=\; 12\left[1 + \frac{\lg\delta^{-1}}{2^{r-j+1}\lg m}\right] \qquad (j = 1, \ldots, r)\ .$$

Our sampler picks h_j at random from $H_{t_j}(2^{r-j}\lg m, 2^{r-j+1}\lg m)$ for $j = 1, \ldots, r$, computes

$$x_i \;=\; (h_1 \circ \cdots \circ h_r)(i)$$

for each $i \in \{0,1\}^{\lg m}$, and outputs $x_1, \ldots, x_m$. To see that this works, we apply Lemma 4.4. Let $\epsilon_* = \epsilon/r$ and let $\delta_* = \delta/r$. Now fix $j \in \{1, \ldots, r\}$. We need to check that

$$2^{2^{r-j}\lg m} \;\geq\; \frac{t_j}{\epsilon_*^2\delta_*^{2/t_j}}\ .$$

Taking logs and noting that $r \leq \lg l$ it suffices to show

$$2^{r-j}\lg m \;\geq\; \lg t_j + 2\lg(\epsilon^{-1}\lg l) + \frac{2}{t_j}\lg(\delta^{-1}\lg l)\ .$$

To establish the above inequality, it suffices to show
(1) $\lg m \geq 3\lg t_j$
(2) $\lg m \geq 6\lg(\epsilon^{-1}\lg l)$
(3) $t_j \lg m \geq 6\lg(\delta^{-1}\lg l)$.
Inequality (1) follows from the fact that $m \geq [12(1+\lg\delta^{-1})]^3 \geq t_j^3$. Inequality (2) follows from $m \geq (\epsilon^{-1}\lg l)^6$. For (3) first note that since $t_j \geq 12$ it suffices to show $2\lg m \geq \lg\delta^{-1} + \lg\lg l$. But this last inequality follows from the fact that the definition of m implies $m \geq \lg\delta^{-1}$ and $m \geq \lg l$. Finally, the number of coin tosses used by the sampler is

$$\begin{aligned}\textstyle\sum_{j=1}^{r} t_j\, 2^{r-j+1}\ \lg m \;&=\; 12\ \lg m \textstyle\sum_{j=1}^{r} 2^{r-j+1}\ + r\ \lg\delta^{-1}\\ &=\; O(2^r\ \lg m)\ + r\ \lg\delta^{-1}\ .\end{aligned}$$

But $2^r = l/\lg m$ and $r \leq \lg l$ so this is $O(l + \log\delta^{-1}\cdot\log l)$ as desired. This concludes the proof. ■

We note that the number of random bits used by the sampler in our theorem is independent of ϵ. The $\log l$ factor in the number of random bits comes from the fact that we have about $O(\log l)$ iterations of the sampling.

5 Application

We prove Theorem 1.2. Our analysis uses the notion of the accepting probability function of an AM game [3, 4], so we begin by recalling it. We then note that we may restrict our attention to a special class of games, provide an overview of the construction, and finally provide the proof.

5.1 Preliminaries

Fix an Arthur-Merlin game. Recall that when the message exchange is over, Arthur decides whether to accept or reject the common input w by evaluating a boolean function $\rho(w, \vec{c}) \in \{0,1\}$, applied to w and the transcript $\vec{c}$ transcript of the conversation. The function ρ, which must be polynomial time computable in the length of its first input, is called Arthur's *decision predicate*. Since Merlin may be assumed to make optimal moves towards the goal of making Arthur accept, the game is fully specified given ρ, the move structure (namely who plays first and the number of moves), and the message lengths. The accepting probability function $acc(\cdot,\cdot)$ captures the interaction of Arthur with such an "optimal Merlin," and is defined by reverse induction on partial conversations, as follows (cf. [3, 4]):

- Let $\vec{c}$ be a conversation. Then the value of the accepting probability function at $\vec{c}$ is the value of the deciding predicate. That is, $acc(w, \vec{c}) = \rho(w, \vec{c})$.
- Let $\vec{x}$ be a partial conversation at a point in the game where the next move y is Merlin's. Then the value of the accepting probability function at $\vec{x}$ is the *maximum* value of all its extensions by the Merlin move y. That is, $acc(w, \vec{x}) = \max_y acc(w, \vec{x}.y)$.
- Let $\vec{x}$ be a partial conversation at a point in the game where the next move r is Arthur's. Then the value of the accepting probability function at $\vec{x}$ is the *average* value of all its extensions by the Arthur move r. That is, $acc(w, \vec{x}) = \mathbf{E}_r\, acc(w, \vec{x}.r)$.

The accepting probability of the game on input w is $acc(w) \overset{\text{def}}{=} acc(w, \lambda)$, where λ is the empty string. The *error probability* of the game on input w with respect to a language L is defined by

$$err_L(w) = \begin{cases} 1 - acc(w) & \text{if } w \in L \\ acc(w) & \text{otherwise.} \end{cases}$$

The error probability of the game, with respect to L, is $e_L\colon \mathsf{N} \to [0,1]$ defined by $e_L(n) = \sup_{|w|=n} err_L(w)$. Thus the game is a proof system for L if $e_L \le 1/3$.

The error probability can be reduced from $\le 1/3$ to $\le 2^{-k}$ by running $O(k)$ independent copies of the given game in parallel and taking a majority vote on the outcomes, for any polynomial $k(n)$. This process, usually called *error reduction*, preserves the number of moves while multiplying the message lengths of both parties by a factor of $O(k)$.

It is convenient to restrict our attention to games in which $l(n) \ge \log n$. This does not reduce the generality of our results, since the following lemma of [5] says that any Arthur-Merlin game can transformed, without increasing the number of rounds, and without increasing the number of coins flipped by more than a constant factor, into one in which Arthur's messages are of length $\ge \log n$.

Lemma 5.1 [5] Suppose we have an Arthur-Merlin proof system for L in which (some or all of) Arthur's messages are of length $\le \log n$. Then we can construct a new Arthur-Merlin proof system for L in which all Arthur's messages are of length $\ge 2\log n$, the total number of moves is no more than before, and the number of coins flipped increases by at most a constant factor.

Note however that the above transformation will increase the length of Merlin's messages by a poly(n) factor.

5.2 Overview of our solution

In this section we give a high level overview of the ideas behind our randomness-efficient speedup theorem. We begin with a discussion of the proof of [4] and then go on to our approach and see how it motivates oblivious sampling.

Following [3, 4] it is convenient to specify the move structure of games symbolically. For example, an $(\mathrm{AM})^g\mathrm{A}$ game is one which consists of $g = g(n)$ rounds —each of these consisting of an Arthur move followed by a Merlin move— followed by a single move by Arthur.

The goal is to convert a game of the form $(\mathrm{AM})^{2g}\mathrm{A}$ into one of the form $(\mathrm{AM})^{g}\mathrm{A}$. In [4], the given game is visualized as $(\mathrm{AMAM})^{g}\mathrm{A}$. Then, in each AMAM block, MAM is converted to AMA. Merging consecutive Arthur moves gives the desired outcome.

Suppose an MAM segment consists of Merlin sending a string y_1, then Arthur sending a (random) $r \in \{0,1\}^{l(n)}$, and finally Merlin responding with y_2. The corresponding AMA game will begin with Arthur first sending random and independent $r^1, \ldots, r^m \in \{0,1\}^{l(n)}$, for an appropriate value of m. Intuitively, Arthur is playing his r move before it is his turn, and so as not to let Merlin gain an advantage, he plays many independent copies of it, forking the game out onto m different "boards." Merlin then responds with y_1 together with $y_2^1, \ldots, y_2^m$. That is, he plays his original y_1 move, and then offers a corresponding response y_2^j to each r^j. Finally, to control the blowup, Arthur selects one of the boards at random by sending a (random) $t \in \{1, \ldots, m\}$. The result of this game segment is then taken to be y_1, r^t, y_2^t and the players proceed to the next block.

Clearly Merlin's chances of winning can only increase in the new game. One has to show they do not increase too much. The analysis of [4] begins by showing that a block is "lucky" with high enough probability (a block is deemed lucky if, given that the value of the accepting probability function at the start of the MAM block was a, the corresponding value in the switched AMA block is $O(ga)$). Their global strategy is now a "greedy" one: they ask that *all* g blocks (top to bottom) be lucky. The $O(g)$ factor blows up to $O(g^g)$ and the result is that for things to work the error probability of the original game (the value of the accepting probability function at the top) must be $\leq O(g^{-g})$. Thus they must begin by reducing the error of the game to this amount, and this error reduction already has an $O(g \log g)$ factor of cost in coins.

The rest of the blowup comes from the number of boards m that must be forked to do a single MAM to AMA switch: their analysis needs m to be $O(g)$ times the *total* number of bits communicated in the (now reduced error) game.

Given that a large factor of the cost is in error reduction, a tempting solution springs to mind. In [5], a randomness-efficient error reduction method is presented. Why not just use this? The reason this will not help is that the second step of the above speedup, namely switching MAM to AMA, has a cost which depends on the length of the messages from Merlin to Arthur in the M steps. The amplification method of [5], while reducing Arthur's coin flips, increases the length of Merlin's messages, and if we were to use this the final cost in randomness of our speedup would actually end up being even worse than that of [4]. This fact, pointed out to us by Goldreich, motivates us to try a different approach.

Our first step, rather than try to reduce the error cheaply, is to remove the need for error reduction. To do this requires stronger properties from the MAM to AMA switch. We will move away from the approach of requiring certain "lucky" blocks, and ask instead for a good "average" property. This will enable us to derive much tighter bounds on the change in the value of the accepting probability function: we will prove that with high probability, the change can be restricted to an additive amount (recall that in [4] the change is a multiplicative factor of $O(g)$). We present this novel MAM to AMA switch as a general Switching Lemma for converting a game of the form X(MAM)Y, (where X, Y are arbitrary game segments) into one of the form X(AMA)Y.

This approach would already serve to yield some improvement over [4], but the main motivation is that the problem has now been cast in a form which is amenable to the use of oblivious sampling. The final idea is to use oblivious sampling in the MAM to AMA switch: Arthur sends a (random) seed of a sampler which is used to specify the sequence of messages $r^1, \ldots, r^m$. The need for the obliviousness is now clear: the estimation must consist of taking the average because such an average is what occurs in the definition of the accepting probability function itself. Note also that the function we will approximate is never explicitly computed by anyone; it occurs only in the proof.

5.3 Switching Lemma and Speedup

The following Switching Lemma is a general tool for converting MAM to AMA in the middle of an arbitrary game. It gives tight bounds on the accepting probability of the new game in terms of the accepting probability of the old one.

In the new game, Arthur will use a universal (l, ϵ, δ)-oblivious sampler S. At a first reading we suggest the reader think of S as just being the sampler given by the standard method; that is, S just outputs $O(\epsilon^{-2} \log \delta^{-1})$ uniformly and independently distributed sample points. This is the route to a simpler proof of the speedup theorem of [4], but one which nonetheless yields a final game of lower complexity than that derived in [4]. To reduce the randomness further, we will use the sampler of Theorem 4.5 in the role of S.

We will write $\mathbf{E}_i\, f(x_i)$ for $\frac{1}{m}\sum_{i=1}^{m} f(x_i)$; because we will have Arthur sending i at random this will be more natural. Below X and Y denote arbitrary game segments.

Lemma 5.2 (Switching Lemma) Suppose we have a game of the form X(MAM)Y in which, in the middle MAM segment, Merlin's messages are of length $q = q(n)$ and Arthur's message is of length $l = l(n)$. Let $\eta = \eta(n)$ be positive with $\eta^{-1} \leq \text{poly}(n)$, and suppose S is a universal $(l, \eta/2, \eta\, 2^{-q-1})$-oblivious sampler which uses $r(n)$ random bits and outputs $m(n)$ sample points. Then we can construct a new game which has the form X(AMA)Y and satisfies:

(1) $|acc^*(w) - acc(w)| < \eta(n)$ for all $w \in \{0,1\}^*$. Namely, the accepting probabilities of the new and the old game, respectively, differ by at most η on any input.

(2) In the AMA segment which replaces the MAM one, Arthur's first message has length $r(n)$ and his second message has length $O(\log n)$, while Merlin's message has length $O(mq)$.

(3) Message lengths in the rest of the game (ie. X, Y) are unchanged.

Proof: Denote the messages of the MAM game by y_1, r, y_2. In the switched game AMA,

- Arthur first sends a random seed s for S, thus using $r(n)$ random bits;
- Merlin responds with $y_1, \vec{y}_2 = (y_2^1, \ldots, y_2^m)$;
- Arthur now sends a random $t \in \{1, \ldots, m\}$, thus using $\log m$ random bits.

We'll denote by $r^1, \ldots, r^m$ the sample points output by S based on the seed s. The deciding predicate of the new game is defined to be

$$\rho^*(w, \vec{x}.sy_1\vec{y}_2t.\vec{z}) \;=\; \rho(w, \vec{x}.y_1r^ty_2^t.\vec{z})\ ,$$

where ρ is the deciding predicate of the old game.

Let $acc^*(\cdot,\cdot)$ and $acc(\cdot,\cdot)$ be the accepting probability functions of the new and old game respectively. It suffices to show that $|acc^*(w,\vec{x}) - acc(w,\vec{x})| < \eta$ for each fixed sequence $\vec{x}$ of moves of the X game segment. So consider $\vec{x}$ fixed. Then

$$\begin{aligned} acc^*(w,\vec{x}) &= \mathbf{E}_s \max_{y_1} \max_{\vec{y}_2} \mathbf{E}_t\, acc^*(w, \vec{x}.sy_1\vec{y}_2t) \\ &= \mathbf{E}_s \max_{y_1} \max_{\vec{y}_2} \mathbf{E}_t\, acc(w, \vec{x}.y_1r^ty_2^t) \\ &= \mathbf{E}_s \max_{y_1} \mathbf{E}_t \max_{y_2} acc(w, \vec{x}.y_1r^ty_2) \\ &= \mathbf{E}_s \max_{y_1} \mathbf{E}_t\, acc(w, \vec{x}.y_1r^t)\ . \end{aligned}$$

So it suffices to show that

$$|\mathbf{E}_s \max_{y_1} \mathbf{E}_t\, acc(w, \vec{x}.y_1r^t) - acc(w,\vec{x})| < \eta\ .$$

We say that s is *good* if

$$|\max_{y_1} \mathbf{E}_t \, acc(w, \vec{x}.y_1 r^t) - acc(w, \vec{x})| < \eta/2 \, .$$

The universal $(l, \eta/2, \eta \, 2^{-q-1})$-oblivious sampler guarantees that for each fixed y_1 we have

$$\Pr\left[\, |\mathbf{E}_t \, acc(w, \vec{x}.y_1 r^t) - acc(w, \vec{x}.y_1)| < \eta/2 \,\right] \;\geq\; 1 - \eta \, 2^{-q-1} \, ,$$

the probability being over the random choice of the seed s. This implies that s is good with probability at least $1 - \eta/2$. Using this we get

$$\begin{aligned} |\mathbf{E}_s \max_{y_1} \mathbf{E}_t \, acc(w, \vec{x}.y_1 r^t) - acc(w, \vec{x})| &\leq \mathbf{E}_s \, |\max_{y_1} \mathbf{E}_t \, acc(w, \vec{x}.y_1 r^t) - acc(w, \vec{x})| \\ &< \eta \, , \end{aligned}$$

as desired. ∎

Repeated applications of the switching lemma to non-overlapping segments of the game yields a speedup theorem in which, in the final game, Arthur's message length depends on the number of coins used by the oblivious sampler and Merlin's message length depends on the number of sample points output by the oblivious sampler.

Theorem 5.3 (Efficient Speedup Theorem) Suppose we are given a $(\mathrm{AM})^{2g}\mathrm{A}$ proof system for L in which Arthur's messages are of length $l = l(n)$ and Merlin's messages are of length $q = q(n)$, and suppose we have a universal $(l, \eta, \eta \, 2^{-q-1})$-oblivious sampler for $\eta(n) = 1/6g(n)$ which uses $r(n)$ random bits and outputs $m(n)$ sample points. Then we can construct a $(\mathrm{AM})^{g}\mathrm{A}$ proof system for L in which Arthur's messages are of length $O(r + \log n)$ and Merlin's messages are of length $O(mq)$.

Proof: By standard error reduction we can assume that the error probability is at most 1/6. We visualize the game as $(\mathrm{AMAM})^{g}\mathrm{A}$ and then apply Lemma 5.2 $g(n)$ times; the i-th application is to the i-th AMAM segment. ∎

Note that we did not need to reduce the error probability of the original game by more than a constant factor.

Let's look at randomness efficiency. We begin by recalling that in the result of [4], Arthur uses $O(lqg^3 \log g)$ random bits per round in the final game. How much do we use? It depends on which oblivious sampler we use, as follows.

We focus on $l \geq \log n$ which by Lemma 5.1 is without loss of generality. So if $r \geq l$ then the number of coins used by Arthur in the final game is $O(r)$. Now let $\epsilon = \eta$ and $\delta = \eta \, 2^{-q-1}$. If we set the sampler in Theorem 5.3 to the standard universal (l, ϵ, δ)-oblivious sampler then, in the final game, Arthur uses $O(l\epsilon^{-2} \log \delta^{-1}) = O(lg^2 \cdot (q + \log g))$ random bits per round, already an improvement over [4]. If instead we set it to the sampler of Theorem 4.5 then Arthur uses $O(l + \log \delta^{-1} \cdot \log l) = O(l + (q + \log g) \cdot \log l)$ random bits per round, which yields Theorem 1.2. Finally, note that if one can show the existence of a universal (l, ϵ, δ)-oblivious sampler which uses only $O(l + \log \delta^{-1})$ random bits and poly(n) sample points then Theorem 5.3 implies that Arthur uses only $O(l + q)$ random bits per round in the final game.

Acknowledgments

We thank Oded Goldreich for pointing out that a randomness-efficient speedup theorem does *not* follow from the existence of a randomness-efficient error-reduction technique and suggesting that we look for a direct proof. We thank Nabil Kahale for drawing our attention to [13].

In the course of this work, the first author was at MIT and later at the IBM T.J. Watson Research Center, New York. The second author was at MIT.

References

[1] W. Aiello, S. Goldwasser and J. Håstad. On the Power of Interaction. *Combinatorica* **10**(1), 3–25, 1990.

[2] M. Ajtai, J. Komlos and E. Szemeredi. Deterministic Simulation in Logspace. Proceedings *of the* 19th *Annual Symposium on Theory of Computing*, ACM, 1987.

[3] L. Babai. Trading Group Theory for Randomness. Proceedings *of the* 17th *Annual Symposium on Theory of Computing*, ACM, 1985.

[4] L. Babai and S. Moran. Arthur-Merlin Games: A Randomized Proof System, and a Hierarchy of Complexity Classes. *J. Computer and System Sciences* **36**, 254–276, 1988.

[5] M. Bellare, O. Goldreich and S. Goldwasser. Randomness in Interactive Proofs. *Computational Complexity* **3**, 319–354, 1993.

[6] M. Bellare and J. Rompel. Randomness-efficient Sampling of Arbitrary Functions. MIT/LCS/TM-433.b, July 1990.

[7] B. Berger and J. Rompel. Simulating $(\log^c n)$-wise independence in NC. Proceedings *of the* 30th *Symposium on Foundations of Computer Science*, IEEE, 1989.

[8] R. Boppana, J. Håstad and S. Zachos. Does co-NP have Short Interactive Proofs? *Info. Processing Letters* **25**, 127–132, 1987.

[9] R. Canetti, G. Even and O. Goldreich. Lower bounds for sampling algorithms for estimating the average. TR-686, Computer Science Dept., Technion, August 1991.

[10] L. Carter and M. Wegman. Universal Classes of Hash Functions. *J. Computer and System Sciences* **18**, 143–154, 1979.

[11] B. Chor and O. Goldreich. On the Power of Two–Point Based Sampling. *J. of Complexity* **5**, 96–106, 1989.

[12] A. Cohen and A. Wigderson. Dispersers, Deterministic Amplification, and Weak Random Sources. Proceedings *of the* 30th *Symposium on Foundations of Computer Science*, IEEE, 1989.

[13] D. Gillman. A Chernoff bound for random walks on expander graphs. Proceedings *of the* 34th *Symposium on Foundations of Computer Science*, IEEE, 1993.

[14] O. Goldreich and A. Wigderson. Tiny families of functions with random properties. Proceedings *of the* 26th *Annual Symposium on Theory of Computing*, ACM, 1994.

[15] S. Goldwasser, S. Micali and C. Rackoff. The Knowledge Complexity of Interactive Proofs. *SIAM J. Computing* **18**(1), 186–208, 1989.

[16] S. Goldwasser and M. Sipser. Private Coins versus Public Coins in Interactive Proof Systems. Proceedings *of the* 18th *Annual Symposium on Theory of Computing*, ACM, 1986.

[17] W. Hoeffding. Probability inequalities for sums of bounded random variables. *J. of the American Statistical Association* **58**, 13–30, 1963.

[18] R. Impagliazzo and D. Zuckerman. How to Recycle Random Bits. Proceedings *of the* 30th *Symposium on Foundations of Computer Science*, IEEE, 1989.

[19] C. Lund, L. Fortnow, H. Karloff and N. Nisan. Algebraic Methods for Interactive Proof Systems. *JACM* **39**(4), 859–868, 1992.

[20] N. Nisan. Pseudorandom Generators for Space Bounded Computation. Proceedings *of the* 22nd *Annual Symposium on Theory of Computing*, ACM, 1990.

[21] N. Nisan and D. Zuckermann. More deterministic simulation in Logspace. Proceedings *of the* 25th *Annual Symposium on Theory of Computing*, ACM, 1993.

[22] J. Rompel. Techniques for computing with low independence randomness. Ph. D Thesis, Dept. of Computer Science, MIT, September 1990.

[23] J. Schmidt, A. Siegel, A. Srinivasan. Chernoff-Hoeffding bounds for applications with limited independence. Proceedings *of the* 4th *Annual Symposium on Discrete Algorithms*, ACM-SIAM, 1993.

[24] A. Shamir. IP = PSPACE. *JACM* **39**(4), 869–877, 1992.

[25] D. Zuckerman. Private communication, September 1994.

A Proofs of LI Tail Inequalities

We prove the low independence tail inequalities of Section 2, beginning with the first. The main lemma in this case is the following bound on the t-th central moment of the sum of *independent* random variables.

Lemma A.1 Let $t \geq 2$ be an even integer. Suppose $Y_1, \ldots, Y_n$ are independent random variables taking values in $[0,1]$. Let $Y = Y_1 + \cdots + Y_n$ and $\mu = \mathbf{E}[Y]$. Then

$$\mathbf{E}[(Y-\mu)^t] \;\leq\; 2e^{\frac{1}{6t}}\sqrt{\pi t}\left(\frac{nt}{e}\right)^{t/2} .$$

Let's apply this to derive Lemma 2.2. First note that since t is even, $|X-\mu|^t = (X-\mu)^t$ and we can drop the absolute values in this expression. Now observe that $(X-\mu)^t$ is a polynomial in $X_1, \ldots, X_n$ of total degree t. So by linearity of expectation, $\mathbf{E}[(X-\mu)^t]$ can be computed under the assumption that $X_1, \ldots, X_n$ are fully independent. Thus $\mathbf{E}[(X-\mu)^t]$ is bounded as indicated by Lemma A.1. Lemma 2.2 follows by an application of Markov's inequality.

We now proceed to prove Lemma A.1. This proof is facilitated by the use of integrals to express the expectation and make the estimates. It also uses standard Chernoff bounds, the Gamma function, and Stirling's approximation. Let us begin by recalling the relevant facts.

A simple, well known expression for the expectation is the following.

Lemma A.2 Let Z be a non-negative real valued random variable. Then $\mathbf{E}[Z] = \int_0^\infty \Pr[Z > x]\, dx$.

Also well known is the following Chernoff-type bound:

Lemma A.3 Suppose $Y_1, \ldots, Y_n$ are independent random variables taking values in $[0,1]$, $Y = Y_1 + \cdots + Y_n$, $\mu = \mathbf{E}[Y]$, and $a \geq 0$. Then

$$\Pr\left[\, |Y - \mu| > a \,\right] \;<\; 2e^{-a^2/2n} \;.$$

We recall that the Gamma function is defined for integer k by

$$\Gamma(k+1) \;=\; \int_0^\infty y^k e^{-y}\, dy \;=\; k\,!$$

Finally, we will use the following form of Sterling's formula which gives an upper bound for all integers k:

$$k\,! \;<\; e^{\frac{1}{12k}} \sqrt{2\pi k} \left(\frac{k}{e}\right)^k \;.$$

We now prove Lemma A.1. By Lemma A.2

$$\begin{aligned} \mathbf{E}[(Y-\mu)^t] &= \int_0^\infty \Pr\left[\, (Y-\mu)^t > x \,\right] dx \\ &= \int_0^\infty \Pr\left[\, |Y-\mu| > x^{1/t} \,\right] dx \;. \end{aligned}$$

By Lemma A.3 this is bounded above by

$$2\int_0^\infty e^{-\frac{x^{2/t}}{2n}}\, dx \;.$$

With the change of variable $y = x^{2/t}/2n$ the integral becomes

$$\begin{aligned} 2 \cdot \frac{t}{2} \cdot (2n)^{t/2} \int_0^\infty y^{\frac{t}{2}-1} e^{-y}\, dy &= 2 \cdot (2n)^{t/2} \cdot \frac{t}{2} \cdot \Gamma\left(\frac{t}{2}\right) \\ &= 2 \cdot (2n)^{t/2} \cdot (t/2)\,! \\ &< 2 \cdot (2n)^{t/2} \cdot e^{\frac{1}{6t}} \sqrt{\pi t} \left(\frac{t}{2e}\right)^{t/2} \\ &= 2e^{\frac{1}{6t}} \sqrt{\pi t} \left(\frac{nt}{e}\right)^{t/2} \;. \end{aligned}$$

This completes the proof of Lemma A.1 and hence of Lemma 2.2.

To prove Lemma 2.3 we proceed similarly. The main lemma is the following.

Lemma A.4 Let $t \geq 2$ be an even integer. Suppose $Y_1, \ldots, Y_n$ are independent random variables taking values in $[0,1]$. Let $Y = Y_1 + \cdots + Y_n$ and $\mu = \mathbf{E}[Y]$. Then

$$\mathbf{E}[(Y-\mu)^t] \;\leq\; 2e^{1/6t}\sqrt{\pi t}\left(\frac{5}{2e}\right)^{t/2} \cdot (t\mu + t^2)^{t/2} \;.$$

Lemma 2.3 is derived from this in the same way that Lemma 2.2 was derived from Lemma A.1.

The proof of Lemma A.4 follows the paradigm of the proof of Lemma A.1. However, we will need a different Chernoff bound, one which permits the kind of integration we want to use later.

Although (stronger) versions of the following lemma appear in the literature [17], the form below is simpler and more convenient for our purposes.

Lemma A.5 Suppose $Y_1, \ldots, Y_n$ are independent random variables taking values in $[0,1]$, $Y = Y_1 + \cdots + Y_n$, $\mu = \mathbf{E}[Y]$, and $a \geq 0$. Then

$$\begin{aligned} \Pr[\, |Y-\mu| > a \,] &< \max(\, 2e^{-3a^2/8\mu} \, , \, e^{-2a/5} \,) \\ &< 2e^{-3a^2/8\mu} + e^{-2a/5} \, . \end{aligned}$$

The reader is referred to [22] for a proof of the above. Now:

$$\begin{aligned} \mathbf{E}[(Y-\mu)^t] &= \int_0^\infty \Pr\left[\, |Y-\mu| > x^{1/t} \,\right] dx \\ &\leq 2\int_0^\infty e^{-\frac{3x^{2/t}}{8\mu}} \, dx \, + \int_0^\infty e^{-\frac{2x^{1/t}}{5}} \, dx \, . \end{aligned} \tag{2}$$

Changing variables to $y = 3x^{2/t}/8\mu$, and then using the definition of the Gamma function and Stirling's approximation as above, the first term of the sum in (2) can be bounded by

$$2 \cdot \frac{t}{2} \cdot \left(\frac{8\mu}{3}\right)^{t/2} \int_0^\infty y^{\frac{t}{2}-1} e^{-y} \, dy \;\; < \;\; 2e^{\frac{1}{6t}} \sqrt{\pi t} \left(\frac{4}{3e}\right)^{t/2} \cdot (t\mu)^{t/2} \, . \tag{3}$$

Similarly with the change of variable $z = 2x^{1/t}/5$, the second term of the sum in (2) can be bounded by

$$e^{\frac{1}{12t}} \sqrt{2\pi t} \cdot \left(\frac{5}{2e}\right)^t \cdot t^t \tag{4}$$

Putting together (2), (3) and (4) concludes the proof of Lemma A.4, and hence of Lemma 2.3.