List-Decoding of Linear Functions and Analysis of a Two-Round Zero-Knowledge Argument

Cynthia Dwork[1], Ronen Shaltiel[2], Adam Smith[3], and Luca Trevisan[4]

[1] Microsoft Research, SVC; 1065 La Avenida Mountain View, CA 94043 USA. dwork@microsoft.com
[2] Department of Computer Science and Applied Math, The Weizmann Institute of Science, Rehovot 76100 Israel. ronens@wisdom.weizmann.ac.il
[3] MIT Computer Science and AI Lab, Cambridge, MA 02139. Work performed while visiting Microsoft Research, SVC. adsmith@mit.edu
[4] University of California, Berkeley. luca@cs.berkeley.edu

Abstract. Dwork and Stockmeyer showed 2-round zero-knowledge proof systems secure against provers which are resource-bounded during the interaction [6]. The resources considered are running time and advice (the amount of precomputed information). We re-cast this construction in the language of list-decoding. This perspective leads to the following improvements:

– We give a new, simpler analysis of the protocol's unconditional security in the advice-bounded case. Like the original, the new analysis is asymptotically tight.
– Dwork and Stockmeyer considered the cases in which the prover is either time-bounded or advice-bounded. We examine the case in which the prover is simultaneously bounded in both these resources, and achieve a substantially better analysis: we prove security (of a slight simplification of their protocol) under the assumption that there exists $g \in DTIME(2^s)$ such that g is hard in the *worst case* for MAM circuits of size $O(2^{s(\frac{1}{2}+\gamma)})$ for some $\gamma > 0$. Here s is the input length and MAM corresponds the class of circuits which are verifiers in a 3-message IP (with constant soundness error) in which the prover sends the first message. In contrast, Dwork and Stockmeyer require a function that is average-case hard for "proof auditors," a new model of computation invented for this purpose. We can base soundness on the worst-case hardness because of a novel non-deterministic list-decoding procedure similar to that of Trevisan and Vadhan [24].

1 Introduction

In this paper we consider 2-round (that is, two-message) zero-knowledge proof systems for NP. Recently, Dwork and Stockmeyer constructed 2-round, black-box, public-coin, zero-knowledge interactive arguments for all of NP, in a model in which the prover is resource-bounded [6].

Two kinds of bounds are considered: on the running time of the prover during the interaction, and on the prover's advice, that is the number of bits of advice the prover may have access to during the interaction. In a little more detail, the prover is split into a preprocessing part and an interaction part. No resource-boundedness is assumed during preprocessing—only the resources used *during* the protocol are limited. In the advice-bounded case, only a bounded amount of information may be passed from the pre-processing part to the interaction part; in the time-bounded case, the running time of the interaction part is bounded. By "bounded", we mean that the resource bounds are *fixed* polynomials in the security parameter.

The Dwork-Stockmeyer (DS) protocol uses as a primitive a linear function f with a certain hardness property. The hardness of f is used to prove the soundness of the protocol against resource-bounded provers. (The specific hardness property varies according to which resources of the prover are bounded; very roughly, f must be hard to compute on random inputs by a circuit with the interacting prover's resources, plus limited non-determinism.) For the case of advice-bounded provers, they show that for each ℓ, if f_ℓ (f restricted to $\{0,1\}^\ell$) is a random linear function from $\{0,1\}^\ell$ to $\{0,1\}^\ell$, then with high probability the chosen f_ℓ will yield a protocol with soundness error $2^{-\ell^{1/d}}$, for any constant $d > 1$. For provers that are time-bounded (and restricted to polynomial advice, but with no specific polynomial bound on advice) they conjecture that a fixed, efficiently computable function f exists that satisfies the appropriate hardness property, but the conjecture is not shown to be implied by standard complexity or cryptographic assumptions, and no candidate for such an f is given.

The goal of this work is a better understanding of the hardness assumptions behind the protocol's soundness. We show that the standard connection between list-decodable error-correcting codes and average-case hardness (see Related Work) holds in this setting. The challenge in applying this connection is that the DS protocol requires *linear* functions—this limits both the kinds of codes one can use and the running time of the list-decoding algorithms. Nonetheless, the connection allows us to give a simpler proof that a random linear function has the required hardness. We also show that a strong, but plausible, complexity-theoretic assumption implies the existence of a fixed function f satisfying the hardness condition needed to make the Dwork-Stockmeyer protocol sound against simultaneously time- and advice-bounded provers.

In the rest of this section, we discuss, informally, the Dwork-Stockmeyer protocol, the connection with coding theory, and our results.

The Dwork-Stockmeyer Protocol

Here is an informal description of the interaction portion of the Dwork-Stockmeyer protocol (see Figure 1). The protocol is based on the existence of a "hard" function f (more on the complexity requirements on f later). Suppose the prover wants to prove that $\tau \in L$, where L is an NP language. Then the verifier sends a random x, and the prover replies with a string β and (roughly) a zap of the statement *"either $\tau \in L$ or β is a valid encryption of $f(x)$."*

Protocol SDS
for language L, using function f which is linear on $GF(2)$,
committing encryption scheme $\mathcal{E}$ that is XOR-malleable, a probabilistic public-key cryptosystem generator $\mathcal{G}$, a zap scheme $\mathcal{Z}$, and constants a, d, e; with inputs τ and w.

0. Before the protocol starts, P does the following precomputation:
   (i) Run $\mathcal{G}(k)$ to produce an encryption key E. Let s be the random string used to produce E.
   (ii) Let $\ell = k^d$ and $x^* \in_R \{0,1\}^\ell$, choose $\alpha \in_R E(x^*)$ and set $\beta = \phi_f(E, \alpha)$. Here $\phi_f(E, \alpha)$ is a uniformly distributed encryption of $f(x^*)$. The existence of a function ϕ_f (that takes an encryption key E and ciphertext $\alpha \in E(x)$ and produces ciphertext $\beta \in E(f(x))$) follows from the linearity of f and malleability of $\mathcal{E}$.
   (Note: the length of the precomputed information, E, s, α, β, is $O(\ell k)$.)
1. $V \longrightarrow P$: V chooses $x \in_R \{0,1\}^\ell$ and an additional string ρ of random bits that will used in zaps, and sends x and ρ to P.
2. $P \longrightarrow V$:
   (i) Send to V: τ, E, α, and β.
   (ii) For using the witness w proving that $\tau \in L$, send the second-round message of a zap that
$$\tau \in L \ \vee \ (E \in \mathcal{G}(k) \wedge \alpha \in E(x)). \qquad (1)$$
3. V accepts iff:
   (i) P responds within time ak^e (time-bounded case only), and
   (ii) $\beta = \phi_f(E, \alpha)$ and
   (iii) the verifier for the zap in (1) accepts.

Fig. 1. Protocol SDS (simplification of [6] protocol).

Intuitively, the protocol is *complete* because the honest prover will just send a random β and will carry out the zap using the witness for $\tau \in L$; the protocol is *simulatable* because the simulator (that is not resource bounded) will give a β that is an encryption of $f(x)$ and will essentially use this as the witness for the zap. The main part of [6] involves an implementation that realizes this informal intuition; we give a precise description of the protocol in the Appendix. The main focus of this paper is the *soundness* proof for the protocol, and the readers can skip the details of the protocol itself if they wish.

Regarding soundness, if $\tau \notin L$, then a cheating prover must be able to compute an encryption of $f(x)$ given a random x, but (still intuitively) this is difficult if f is computationally hard and the prover is resource-bounded. A number of technical issues arise in formalizing the intuition above; for example, it is not clear that if f is computationally hard then producing an encryption of f is also computationally hard. Finally, it remains to find the right complexity measure for f which makes the analysis possible.

To this end, Dwork and Stockmeyer introduce the notion of a *proof auditor*.[1] A proof auditor is an abstract computational model that, roughly speaking, is a

[1] The proof auditor is an imaginary device used in the *analysis* of the protocol, it is not *part* of the protocol.

randomized and non-uniform version of $NP \cap coNP$. The analysis in [6] shows that a prover that successfully cheats with probability χ can be converted into a proof auditor of similar complexity (that is, advice size and running time) that computes f on roughly a χ fraction of inputs.

Hence, for the protocol to be sound, one must use functions f that are hard on average against proof auditors of bounded complexity. For completeness, there is another requirement: one should be able, in polynomial time, to compute an encryption of $f(x)$ given an encryption of x. This is possible if f is a linear function over $GF(2)$ and if the encryption scheme is *XOR malleable.* [2] The Goldwasser-Micali cryptosystem, based on the quadratic residuosity assumption, is XOR-malleable.

In summary, the function f to be used in the Dwork-Stockmeyer protocol should be a linear function over $GF(2)$ and should be hard on average against resource-bounded proof auditors. If we want the protocol to be sound against advice-bounded provers (with no running time restriction), then f has to be hard against advice-bounded proof auditors (with no running time restriction). If we want the protocol to be sound only against time-bounded provers (with an advice bound also implied by the time bound), then f has to be hard against time-bounded proof auditors.

Dwork and Stockmeyer [6] give a complicated proof that a random linear function is hard against advice-bounded proof auditors, and they conjecture that there are explicit functions that are hard against time-bounded proof auditors, but they give no such explicit construction based on other complexity assumptions.

Figure 1 gives a more precise description of the DS protocol. (In fact, the version here is somewhat simplified from the original one.) Because the focus of this paper is on the assumptions behind the protocol's soundness, we refer the reader to [6] or to the full version of this paper for more details on the protocol itself.

Our Results

Advice-bounded Proof Auditors Our first result is a connection between list-decodable codes and hardness against advice-bounded proof auditors (of arbitrary running time). We show that if we fix any error-correcting code with good combinatorial list-decoding properties[3], and we pick a random codeword c from the code and let f be the function whose truth-table is c, then with high probability, f is very hard on average against proof auditors of bounded advice. (This is very similar to Trevisan's proof [23] that a random member of a list-decodable code is average-case hard for small circuits with high probability.)

We also show that the set of all linear functions has reasonably good list-decoding properties. It follows that a random linear function is hard on average

[2] A 1-bit encryption scheme is XOR-malleable if one can create an encryption of $a \oplus b$ from encryptions of a and b. The value of malleable encryption schemes was first noted by Rivest, Adleman and Dertouzos [18].

[3] That is, a code such that every sphere of bounded radius contains few codewords.

against advice-bounded proof auditors, and there exist linear functions f for which the Dwork-Stockmeyer protocol is unconditionally sound.[4] Dwork and Stockmeyer had already proved that random linear functions are hard for advice-bounded proof auditors, but our proof is simpler, and it seems to get to the heart of what makes their protocol sound.

Adding Time-Boundedness Next, we turn to proof auditors that are simultaneously time- and advice-bounded. We show how to construct an explicit hard function starting from more standard complexity-theoretic assumptions.

Roughly speaking, we start from a function g that is hard *in the worst case* against a certain type of sub-exponential non-deterministic circuit. We view the truth-table of g as a matrix A, and we define f to be the linear mapping $x \mapsto Ax$. We then show that if there is a proof auditor that can compute f well on average, then there is a non-deterministic circuit that can reconstruct A, and therefore g, violating g's hardness assumption. This analysis can be seen as an algorithmic version of our results that linear functions have good combinatorial list-decoding properties: here we do the list-decoding explicitly, using non-uniformity to choose from the list, and using non-determinism to help with the decoding.

Specifically, we prove security under the assumption that there exists $g \in DTIME(2^s)$ such that g is hard in the *worst case* for MAM circuits of size $O(2^{s(\frac{1}{2}+\gamma)})$ for some $\gamma > 0$. Here s is the input length and MAM corresponds the class of circuits which are verifiers in a 3-message IP (with constant soundness error) in which the prover sends the first message.

Challenges of List-Decoding Linear Functions The use of list-decoding to construct hard-on-average functions is not new (see Related Work). However, the fact that we need hard *linear* functions adds challenges which are the focus of the results described above. First of all, the code itself must be a sub-code of the set of all linear functions. More importantly, there is very little room for play in the hardness assumptions. Any linear function can be computed exactly by a circuit of size and time $O(\ell^2)$ on inputs of length ℓ. This means there is at most a quadratic gap between the resources required to remember a single pair $x, f(x)$ and the resources required to cheat in the [6] protocol. This, in turn, means that the reductions we give (i.e. the list-decoding algorithms) must take much less than quadratic time. For this reason we cannot use standard list-decoding techniques and complexity reductions, since those typically involve polynomial blow-ups.

Non-linear functions It is an open question whether *completely malleable* encryption systems exist. By a completely malleable encryption system we mean that given the encryption of x and a circuit C, one can compute an encryption of $C(x)$ (malleable shemes were originally called *privacy homomorphisms* by Rivest, Adleman and Dertouzos [18]).

[4] Alternatively, this non-explicit construction can be replaced by a preprocessing phase in which V (or a trusted party) randomly chooses such a function and announces it.

If semantically secure, completely malleable encryption is possible, then one does not need f to be linear in the Dwork-Stockmeyer protocol. This avoids the difficulties described above. In particular, one can use Reed-Solomon codes instead of linear functions in the case of advice-bounded proof auditors, and a different (more standard) transformation of a worst-case hard function g into an average case hard function f in the time-bounded case. This leads to a larger (arbitrary polynomial) gap between the resources of an honest prover and those required to cheat. However, the assumption of a completely malleable cryptosystem seems very strong; no candidate is known.

Related Work The work of Dwork and Stockmeyer followed a long line of work on protocols whose participants have bounded computational power and/or bounded communication; we refer the reader to [6] for a discussion. We focus here on the origins of the techniques we use and on previous uses of derandomization in cryptography.

Derandomization Tools in Cryptography: The mathematical tools used in derandomization, such as error-correcting codes, limited independence, expander graphs, and list-decoding, have been used in cryptography for a long time, a prime example being Goldreich-Levin hard bits [7]. There has been a recent explosion of work in cryptography using these tools more explicitly—see, for example, the work of Lu [14] and later Vadhan [25] improving encryption protocols for Maurer's bounded storage model [16, 1] (the work of Lu partly inspired this work). The most closely related work to ours is that of Barak, Ong and Vadhan [2]. By de-randomizing the 2-round zap construction in [5], Barak et al. obtained *uniform* non-interactive witness-indistinguishable proofs and arguments ([5] shows the existence of a non-uniform noninteractive protocol). As in our work on the simultaneously advice- and time-bounded case, [2] base security of a cryptographic protocol on an assumed *worst-case* circuit lower bound.

List-Decoding and Average-Case Hardness: The main technique which we take from the derandomization literature is the connection between list-decoding and average-case hardness. The connection had been part of the oral tradition of the community since the early 1990s, due to the independent observation by Impagliazzo and Sudan that the result of [7] could be interpreted as a list-decoding algorithm for the Hadamard code and that other list-decodable codes could be used to prove similar results.

More specifically, our proof that linear functions form a combinatorially good list-decodable code relies on a lemma of Chor and Goldreich [4] on list-decoding punctured Hadamard codes. In the reduction of Section 4, we need to show that the problem of list-decoding a certain code with certain parameters can be solved in quasi-linear "MAM-time." This result is inspired by a reduction in [24], that also involves very efficient list-decoding algorithms that are sped-up using non-determinitism (actually, in [24], list-decoding is performed by circuits with Σ_i gates, for various i).

Finally, the results on non-linear functions rely on the techniques of Trevisan and Vadhan [24] just mentioned, and also on the techniques of [13, 17, 19] which gave also hardness results for non-deterministic circuits.

2 Resource-Bounded Provers and Proof Auditors

As discussed above, Dwork and Stockmeyer reduced the soundness of their protocol to the existence of linear functions which are hard for *i.o. proof auditors* with bounded resources (recall that completeness and zero-knowledge follow from more standard assumtpions). In this section we collect the results we will need from [6]. First, we a precise definition of proof auditors and state the reduction from [6]; we conclude with the statement of their result on the hardness of linear functions for advice-bounded auditors.

In our discussion, we emphasize that the proof auditors coming from the reduction can be made *single-valued*, a property which we will use in the sequel.

Definition 2.1 (i.o. proof auditors). *An* i.o. proof auditor for function f *is a randomized nondeterministic device. In order to bound the non-uniformity involved, we fix a universal Turing machine UTM which takes an advice string $p \in \{0,1\}^a$. Let $\mathcal{A}$ denote the circuit corresponding to the behaviour of the universal machine on advice string p. That is, for any input $\omega \in \{0,1\}^*$, we say $\mathcal{A}(\omega) = UTM(p, \omega)$.*

The circuit $\mathcal{A}$ takes an input $x \in \{0,1\}^\ell$, as well as a random input $r \in \{0,1\}^R$ and a non-deterministic input z, and outputs a pair $(b, v) \in \{0,1\} \times \{0,1\}^\ell$. We say that an i.o. proof auditor has agreement ϵ with a function f if for infinitely many values $\ell \in \mathbf{N}$:

$$\Pr_{x\in\{0,1\}^\ell, r\in\{0,1\}^R}\left[\forall y \left(\exists z \in \{0,1\}^N (\mathcal{A}(x,r,z) = (1,y)) \iff y = f(x)\right)\right] \geq \epsilon(\ell)$$

The important parameters of an auditor are its advice bound a, its randomness bound R, its non-determinism bound N, its success probability ϵ, and its running time T. Here "i.o." stands for infinitely often (over $\ell \in \mathbf{N}$).

An i.o. proof auditor $\mathcal{A}$ is said to be single-valued everywhere *if for any fixed input x and sequence of coin tosses r, there is at most one value $y \overset{\text{def}}{=} \tilde{f}_{\mathcal{A}}(x, r)$ for which there exists a string z such that $\mathcal{A}(x, r, z) = (1, y)$.*

In other words, an single-valued i.o. proof auditor ϵ-approximates f if there is an ϵ fraction of the (input,coins) pairs (x, r) on which the auditor outputs a *unique* $y = f(x)$. A given circuit $\mathcal{A}$ can ϵ-approximate several different functions.

Theorem 2.1 (Dwork, Stockmeyer [6]) *Let P^* be a cheating prover which is limited, during the protocol, to advice bound $A^*(k)$, time bound $T^*(k)$, and randomness $R^*(k)$. Let $\epsilon^*(k)$ denote P^*'s probability of cheating successfully. There exist constants c_1, c_2, c_3, e such that there is a* single valued *i.o. proof auditor for f having the following bounds, where $k = \lfloor \ell^{1/d} \rfloor$ for a constant d appearing in the description of the protocol:*

$$\begin{aligned}
\textit{advice } a(k) &\leq A^*(k) \\
\textit{running time } T(k) &\leq T^*(k) + O(k^e + \ell k^{c_1} + k^{c_2}) \\
\textit{non-determinism } N(k) &\leq O(k^{c_3}) \\
\textit{randomness } R(k) &\leq R^*(k) \\
\textit{agreement probability } \epsilon(k) &\leq \epsilon^*(k) - 2 \cdot 4^{-k}
\end{aligned}$$

Thus, the DS protocol for f is sound against a certain class of cheating provers if f has no proof auditors from (roughly) the same class with non-negligible agreement.

One of the main results of [6] shows that appropriately "hard" linear functions f exist for the advice bounded case—hence, no special assumptions are necessary beyond the XOR malleable cryptosystem (which can be based on standard number theoretic assumptions), and the existence of trapdoor permutations. [5]

Theorem 2.2 (Random linear functions, [6]) *With probability $1-\delta$, a random linear function $f : \{0,1\}^\ell \to \{0,1\}^\ell$ has no proof auditor with success rate ϵ and advice bound $a = \ell^2 - 3\log\frac{1}{\epsilon} + \log\delta$.*

3 Advice-bounded Proof Auditors

In this section we show that a random codeword from a list-decodable code defines a hard function for advice-bounded proof auditors, and we show that linear functions have good list-decodability properties. These two results imply that random linear functions are hard for advice-bounded proof auditors.

Definition 3.1 (List-decodable code). *Let Σ be a finite alphabet. An injective mapping $C : \{0,1\}^n \to \Sigma^L$ is an $(\epsilon, t(\epsilon))$ list-decodable code if for all $\epsilon > 0$ and all $u \in \Sigma^L$ (u need not be in the image of C), there are fewer than $t(\epsilon)$ codewords (i.e., elements of the image) at Hamming distance $L(1-\epsilon)$ or less from u.*

We are interested in codes which support list-decoding when almost all of the positions in a codeword have been corrupted. We think of elements in Σ^L as functions mapping $\{0, 1, \ldots, L-1\}$ to elements of Σ in the following natural way: for $0 \le i < L$, i is mapped to the ith element in the L-tuple. Let v be a codeword. For all functions $g : \{0, 1, \ldots, L-1\} \to \Sigma$, g has agreement ϵ with v if and only if g is within distance $(1-\epsilon)L$ of v.

We begin with some intuition. Suppose that an auditor is a deterministic machine restricted to a bits of advice (think of the advice as a description of the auditor, to be fed to a universal Turing machine). Suppose we also have a code such that in any ball of relative radius $1-\epsilon$ there are at most $t = t(\epsilon)$ codewords. The number of codewords that have agreement ϵ with some auditor is then at most $t2^a$. If we have 2^n codewords and we pick one of them at random, then the probability that we pick a codeword having agreement ϵ with some auditor is at most $t2^a/2^n$. This intuition does not quite suffice, since we are actually using two different notions of agreement: agreement of a function (g) with the function (defined by a codeword v), and agreement of a proof auditor with a function. In the first case, the notion of agreement is over choices of inputs to

[5] Although the security of the Goldwasser-Micali cryptosystem implies the existence of trapdoor permutations based on factoring, we have no reason to assume that this will be true of all XOR-malleable systems.

the function: two functions f, g have agreement ϵ if the probability over inputs x that $f(x) = g(x)$ is ϵ. In the second case, the probability is also over random coin tosses made by the auditor (see Definition 2.1).

In the proof of the theorem below, we use the list-decoding property, which talks about agreement of the first kind, to bound the number of codewords with which an auditor can have agreement of the second kind.

Theorem 3.1 *Let $C : \{0,1\}^n \to \Sigma^L$ be a list-decodable error-correcting code, with $L = 2^d$, $\Sigma = \{0,1\}^m$, and list size $t(\epsilon)$. Let $c \in_R C$, and let $f : \{0,1\}^d \to \{0,1\}^m$ be given by $f(i) = c_i$. With probability $1 - \delta$, there is no single-valued i.o. proof auditor for f with advice bound $a = n - \log t(\epsilon^2/4) + \log(\epsilon/2) + \log(\delta)$ and which has agreement ϵ or more with f.* [6]

Proof. Recall that we describe an auditor as an input to a universal Turing machine. Consider a particular auditor $\mathcal{A} = UTM(p, \cdot)$, where $|p| = a$. We may define a second auditor, $\mathcal{A}'$, that has no nondeterministic inputs, as follows. On input (x, r), $\mathcal{A}'$ tries all possible values for z to see if there is a unique y such that $\mathcal{A}(x, r, z) = (1, y)$ as z varies. If no such y exists, then $\mathcal{A}'(x, r)$ outputs $(0, \perp)$. If such a y exists, then $\mathcal{A}'(x, r)$ outputs $(1, y)$. Note that, by construction of $\mathcal{A}'$, $\forall f$

$$\Pr_{(x,r)}[\mathcal{A}'(x,r) = (1, f(x))] = \Pr_{(x,r)}[\exists z(\mathcal{A}(x,r,z) = (1,y)) \iff y = f(x)]. \quad (2)$$

Thus, for all ϵ and all functions f, $\mathcal{A}$ is a single valued ϵ-auditor for f if and only if $\mathcal{A}'$ is an ϵ-auditor for f.

At this point, it may be that for any given x, there may exist many r', y' such that $\mathcal{A}'(x, r') = (1, y')$. We wish to restrict our attention to those values y' that occur with sufficient support among the choices for r. To this end, we define a third auditor, $\mathcal{A}''$: On input (x, r), $\mathcal{A}''$ runs $\mathcal{A}'(x, r)$ to obtain (b, y). If $b = 0$, then $\mathcal{A}''(x, r)$ outputs $(0, \perp)$. If $b = 1$, then $\mathcal{A}''(x, r)$ tries enough of the possible choices for r necessary to see if, for at least an $\epsilon/2$ fraction of the r's, $\mathcal{A}'(x, r) = (1, y)$. If so, $\mathcal{A}''(x, r)$ outputs $(1, y)$; otherwise it outputs $(0, \perp)$. $\mathcal{A}''$ has the property that, on any particular x, different values of r can give rise to at most $2/\epsilon$ different values of $\mathcal{A}''(x, r)$.

Lemma 3.2 *If $\mathcal{A}'$ is an ϵ-auditor for f, then $\mathcal{A}''$ is an $\epsilon/2$-auditor for f.*

Proof. For a particular function f, let $W_f(x)$ denote the fraction of random inputs r such that $\mathcal{A}'(x, r) = (1, f(x))$. We know that the expected value (over choice of x) of $W_f(x)$ is the probability (over x and r) with which $\mathcal{A}'$ agrees with f, that is

$$\mathcal{E}_x[W_f(x)] = \Pr_{x,r}[\mathcal{A}'(x,r) = (1, f(x))] \geq \epsilon. \quad (3)$$

[6] The parameter m does not appear explicitly in the proof of Theorem 3.1. In fact, m affects the function $t(\epsilon)$ in the definition of a list-decodable code. The proof never needs specific values for this function.

By construction, $\mathcal{A}''(x,r) = (1, f(x))$ precisely when both $\mathcal{A}'(x,r) = (1, f(x))$ and $W_f(x) \geq \epsilon/2$, so that

$$\Pr_{x,r}[\mathcal{A}''(x,r) = (1, f(x))] = \mathcal{E}_x[W_f(x)|W_f(x) \geq \epsilon/2] \cdot \Pr_x[W_f(x) \geq \epsilon/2].$$

Hence we can write:

$$\begin{aligned}
&\epsilon \leq \mathcal{E}_x[W_f(x)] \\
&= \Pr_{x,r}[\mathcal{A}''(x,r) = (1, f(x))] + \mathcal{E}_x[W_f(x)|W_f(x) < \epsilon/2] \cdot \Pr_x[W_f(x) < \epsilon/2] \\
&\leq \Pr_{x,r}[\mathcal{A}''(x,r) = (1, f(x))] + \mathcal{E}_x[W_f(x)|W_f(x) < \epsilon/2]
\end{aligned}$$

The second term in the last sum can be at most $\epsilon/2$, since we condition on the fact that $W_f(x) < \epsilon/2$. Thus, the probability (over x and r) with which $\mathcal{A}''$ agrees with f must be at least $\epsilon/2$. □

To conclude the proof of Theorem 3.1, we let $J = \lfloor 2/\epsilon \rfloor$, and, for each x, choose values $g_1(x), ..., g_J(x)$ so that $\{g_1(x), ..., g_J(x)\} = \{y : \exists r\ \mathcal{A}''(x,r) = (1, y)\}$. The probability (over x and r) with which $\mathcal{A}$ agrees with f is at most the sum of the agreements of f with the functions $g_1(\cdot), ..., g_J(\cdot)$. Assuming $\mathcal{A}''$ is an $\epsilon/2$ auditor for f, there must be some $i \in [J]$ such that f has agreement $\frac{\epsilon}{2J} = \epsilon^2/4$ with g_i. Thus, the total number of functions f for which the original $\mathcal{A}$ is an ϵ auditor is at most $J \cdot t(\epsilon^2/4) = \frac{2}{\epsilon} t(\epsilon^2/4)$. If describing the auditor requires only a bits of advice, we can describe all the functions which have ϵ-auditors with advice bound a using $a + \log t(\epsilon^2/4) + \log J$ bits. Since there are 2^n codewords, choosing one at random will yield a function with an ϵ-auditor with probability at most $(2^{a+\log t(\epsilon^2/4)+\log(\epsilon/2)})/2^n = \delta$. Thus, choosing a codeword at random yields a function *not* having an ϵ-auditor with advice bound a with probability at least $1 - \delta$. □

Now let C be the set of all linear functions mapping ℓ bits to ℓ bits. Each element of C can be described by ℓ^2 bits, so $C : \{0,1\}^{\ell^2} \to (\{0,1\}^\ell)^{2^\ell}$, and C is injective. C is a code; recall that we think of the codewords (points in the image) as functions. The ith element in the 2^ℓ tuple is the value assigned to i, for $0 \leq i < 2^\ell$. Letting $\Sigma = \{0,1\}^\ell$ and let $L = 2^\ell$, we have that $C : \{0,1\}^{\ell^2} \to \Sigma^L$.

Proposition 3.3 *The code C has list-size $t(\epsilon) = 2^{2.7\ell(\log \frac{1}{\epsilon} + \frac{4}{3})}$.*

Proof. (Sketch.) For any $v \in \Sigma^L$ and any $x \in \{0,1\}^\ell$, we write $v(x)$ to denote the value of v applied to x (recall, words in Σ^L are functions). To prove the Proposition, it suffices to demonstrate how to describe any $\ell \times \ell$ matrix A having agreement ϵ with v using only $2.7\ell(\log \frac{1}{\epsilon} + 4/3)$ bits. Let $a_1, ..., a_\ell \in \{0,1\}^\ell$ denote the rows of A, and $v_i(x)$ denote the i^{th} bit of $v(x)$.

Fact 3.4 (Chor, Goldreich [4]) *If $S \subseteq \{0,1\}^\ell$ has at least $\epsilon \cdot 2^\ell$ elements, and $g : S \to \{0,1\}$ is an arbitrary function, then there are at most $9/\epsilon$ vectors $a \in \{0,1\}^\ell$ such that $Pr_{x \in S}[g(x) = a \cdot x] > 2/3$, where $a \cdot x$ denotes the inner product of a and x.*

Let E_i denote the event that Ax and $v(x)$ agree in the ith bit, for $x \in \{0,1\}^\ell$ (that is, $a_i \cdot x = v_i(x)$). We have

$$\epsilon \leq \Pr_x[E_1 \cdots E_\ell] = \Pr_x[E_1] \cdot \Pr_x[E_2|E_1] \cdots \Pr_x[E_\ell|E_1 \cdots E_{\ell-1}].$$

We first note that at most $\log_{3/2}(1/\epsilon) < 1.7\log(1/\epsilon)$ terms in this product can be smaller than 2/3. To describe the corresponding "bad" rows of A, we specify a_i explicitly, using a total of at most $1.7\ell \log \frac{1}{\epsilon}$ bits.

Now for each of the remaining "good" rows, we have $\Pr[a_i \cdot x = v_i(x)|E_1 \cdots E_{i-1}] \geq 2/3$. Letting $S_i = \left\{x \in \{0,1\}^\ell | E_1 \wedge \cdots \wedge E_{i-1}\right\}$, we can apply Fact 3.4 to see that each such "good" a_i requires only $\log(9/\epsilon) \leq \log(1/\epsilon) + 4$ bits to specify (given the description of the previous ones). Hence, the total number of bits required to describe A is $1.7\ell \log \frac{1}{\epsilon} + \ell(\log \frac{1}{\epsilon} + 4)$. □

The result of [6] on advice bounded provers is now a corollary to Proposition 3.3 and Theorem 3.1[7]. In the next section, we address the non-constructive nature of these results.

Corollary 3.5 ([6], Theorem 7.8 on advice-bounded provers) *There exists a family of linear functions $\{f_\ell\}_{\ell \in \mathbb{N}}$, such that the Dwork-Stockmeyer proof system has soundness error at most $\epsilon + 2 \cdot 4^{-k}$ against provers who are limited to $\ell^2 - 6\ell \log \frac{1}{\epsilon}$ bits of advice (for all $\epsilon < 1/32$).*

4 Simultaneously Time- and Advice-Bounded Provers

We now turn our attention to the case of provers that are simultaneously time- and advice-bounded during the execution of the protocol. We show how to *construct* efficiently decodable linear functions f that have no simultaneously time- and advice- bounded auditors, based on the rather natural assumption that there exist functions $g : \{0,1\}^s \to \{0,1\}$ computable in time $2^{O(s)}$ with no MAM circuits (defined below) of size $O(2^{s(\frac{1}{2}+\gamma)})$, for some $\gamma > 0$. We create a matrix for the linear function by setting its entries to be the truth table of a suitably hard function, call it g. This hard function may have a very short description. The role of the advice bound is again to prevent a cheating prover from bringing the entire matrix of the linear function into the interaction; however, now the prover may be able to bring in the short description of the hard function g, from which the linear function is constructed. It is the time bound, together with the assumed hardness of g, that prevents a cheating prover from computing the entries in the matrix during the course of the execution of the protocol.

One can view the results of this section as an algorithmic version of the results of the previous section: not only are there few linear functions in any given ball

[7] The bound in [6] is slightly stronger: the 6 is replaced by 3. Another proof can be obtained using results of Shpilka giving a set of $2^{\ell^2 - 2\ell \log \frac{1}{\delta}}$ linear functions which have pairwise relative distance $1 - \delta^2/4$ [20]. Yet another possible approach comes from the results of Mansour et al. [15] on universal hash families. Unfortunately, they are too general to yield tight bounds for the case of binary linear maps.

of bounded radius, but given the ball, some extra advice, and non-determinism, each of these linear functions is easy to compute.

The basic schema for our proof comes from the literature on derandomization. Let A be an $\ell \times \ell$ matrix for a linear function mapping $\{0,1\}^\ell$ to $\{0,1\}^\ell$. Let $\tilde{f} : \{0,1\}^\ell \to \{0,1\}^\ell$ be any (not necessarily linear) function having agreement ϵ with A. Then, given a description of $\tilde{f}$, we can describe A using only $\log t(\epsilon)$ additional bits. This is because, by Proposition 3.3, the linear functions form a list-decodable code with codewords in $(\{0,1\}^\ell)^{2^\ell}$, and f can be represented as a string in $(\{0,1\}^\ell)^{2^\ell}$.

This means that, given a circuit $\mathcal{C}$ for $\tilde{f}$, we can describe A using at most $|\mathcal{C}| + \log t(\epsilon)$ bits, where $|\mathcal{C}|$ denotes the size of $\mathcal{C}$. We now wish to consider situations in which this short description of A is in fact a circuit for computing the bits of A.

Suppose that we have an extremely efficient decoding procedure. That is, suppose that given

(a) a circuit $\tilde{\mathcal{C}}$ that has agreement ϵ with a codeword given by a matrix A, and
(b) $\ell^{1+o(1)}$ additional bits of advice (say, to specify A completely),

we can construct a circuit $\tilde{\mathcal{C}}$ which, on inputs i, j, outputs $A_{i,j}$ in time $\ell^{1+o(1)}$, and using $O(1)$ calls to $\tilde{\mathcal{C}}$. Then the existence of a "small" circuit $\tilde{\mathcal{C}}$ which correctly computes the linear map $x \mapsto Ax$ with probability ϵ implies the existence of a "small" circuit $\mathcal{C}$ which, on inputs i, j, computes $A_{i,j}$, where "small" means size $O(\ell^{(1+\gamma)})$ for some constant $\gamma > 0$. Thus, if we use the *truth table* of a hard function g — one which can't be computed using "small" circuits — to provide the bits of the matrix A, then we know that no small circuit can have agreement ϵ with the linear map $x \mapsto Ax$.

Our application of this schema differs from the derandomization literature in a few respects. First and most importantly, we are interested in list-decoding linear functions, and in making sure that the reductions preserve circuits which use less than quadratic resources (we end up with quasi-linear reductions). This means that we cannot use many of the standard results; the reduction in rounds of MAM protocols, or the deterministic list-decoding algorithms for Hadamard codes both require too much time. Perhaps a less important difference is that the circuits we input and output are randomized and non-deterministic, and in fact we output a verifier for a proof system rather than an ordinary circuit (such reductions have been considered before, e.g. [24]). Finally, the result of the reduction is a proof of security for a cryptographic protocol (we know of only one other such cryptographic use of derandomization [2]). We obtain:

Theorem 4.1 *Suppose there exists a function $g : \{0,1\}^s \to \{0,1\}$ that is in $E = DTIME(2^{O(s)})$, but which has no MAM verifier circuits of size $2^{s(\frac{1}{2}+\gamma)}$, for some constant $\gamma > 0$. Then, if XOR-malleable cryptosystems exist, for any constant γ' such that $0 < \gamma' < 2\gamma$, there is a uniformly constructible Dwork-Stockmeyer proof system with negligible error probability against provers with advice and computation time bounded by $O(\ell^{1+\gamma'})$, where ℓ is the input/output* *length of the public function f, and $k = \ell^{\Theta(1)}$ is the security parameter for the encryptions and zaps.*

In order to prove Theorem 4.1, we first give our result on list-decoding of linear functions, where the list-decoding circuits we construct are in fact verifiers for an MAM proof system. We defer the proof of Theorem 4.1 to the end of this section.

Theorem 4.2 *Let F be a field of size $q = 2^{\lceil \log 10/\epsilon^2 \rceil}$. Let $\mathcal{A}$ be a single-valued i.o. proof auditor with non-uniformity bound a, randomness bound R, time bound T, and non-determinism bound N (see Definition 2.1). Let $\tilde{f}_{\mathcal{A}} : F^{\ell'} \times \{0,1\}^R \to F$ be the function computed by $\mathcal{A}$, that is $\tilde{f}_{\mathcal{A}}(x, r)$ is the single value that $\mathcal{A}$ may output on inputs x, r. Suppose that $\tilde{f}_{\mathcal{A}}$ agrees with a linear function given by vector $v \in F^{\ell'}$ with probability at least ϵ, in the following sense:*

$$\Pr_{x,r}[\tilde{f}_{\mathcal{A}}(x, r) = v \cdot x] \geq \epsilon.$$

Then we can construct a verifier circuit Arthur for an MAM protocol which computes the row vector $v \in F^{\ell'}$ with probability at least 2/3. The circuit uses $O(a + \log \ell' \log(1/\epsilon))$ bits of non-uniform advice (some of these are necessary just to have v well-specified), and communication $O(\ell' \log \frac{1}{\epsilon} + R + N)$ (this corresponds to non-deterministic advice from Merlin). The running time of the circuit for Arthur is $\tilde{O}(T + \ell' \log \frac{1}{\epsilon} + R + N)$.

Proof (of Theorem 4.2). There are three phases to the reduction. First, use non-determinism (i.e., the first message from the prover Merlin to verifier Arthur) to guess a candidate vector v'. Next, we use a non-deterministic counting technique, due to Stockmeyer [21], to verify that the candidate v' has agreement close to ϵ (specifically, $\epsilon/2$) with $\tilde{f}$. By Lemma 4.3, there are at most $O(1/\epsilon)$ such vectors. Finally, we start the verifier with a few bits of advice, enabling it to perform a test which, among those close vectors, is passed only by v. The remainder of the advice bits are used as advice in the calls to $\mathcal{A}$. We now describe the details of the agreement test and the selection of v using short advice.

Agreement Test. Let $S \subseteq F^{\ell'} \times \{0,1\}^R$ be the set of pairs (x, r) such that $\tilde{f}_{\mathcal{A}}(x, r) = v' \cdot x$. We wish to verify that $|S| \geq \epsilon q^{\ell'} 2^R$. In fact, we only test that $|S| \geq (\epsilon/2) q^{\ell'} 2^R$.

To do this, the verifier chooses a pairwise-independent random sample U of size M/ϵ from the set $\mathcal{D} = F^{\ell'} \times \{0,1\}^R$, where M is some large constant. Consider the set $U \cap S$. One can choose M appropriately so that when $|S| \leq \epsilon q^{\ell'}/2$, the probability of there being more than $3M/4$ points is at most 1/3. Conversely, when $|S| \geq \epsilon q^{\ell'}$ that same probability is at least 2/3. Thus, to allow Arthur to check that each of the points is also in S, the prover need only send the verifier $3M/4$ points (x, r) in U, together with the non-deterministic inputs z used by $\mathcal{A}$ to produce an output on the input pairs (x, r) (a different z for each pair).

For representing the sample U and verifying membership efficiently, view $\mathcal{D}$ as the field $GF(2^{\ell' \log q + R})$. Then to choose U, the verifier need only choose 2 elements $\alpha, \beta \in \mathcal{D}$ at random. To specify an element of U, a string of $\log \frac{1}{\epsilon} + O(1)$ bits suffices, and it only takes time $\tilde{O}(\ell' \log q + R)$ to reconstruct the full representation (this is the time needed for multiplication in $\mathcal{D}$).

Using Short Advice to Select v. It remains to give a short test to determine whether a given v', having agreement at least $\epsilon/2$, is the correct v'. Let $e = \log(10/\epsilon^2)$. We view v' as a string of $\ell' \log 10/\epsilon^2 = \ell' e$ bits, and apply a standard polynomial fingerprinting scheme.

Namely, choose $p = O(\ell' \log(1/\epsilon) \cdot \frac{1}{\epsilon^2})$, such that p is a power of 2, and work in $GF(p)$. For any string $a \in \{0,1\}^{\ell' e}$, write a as a sequence of $\frac{\ell' e}{\log p}$ elements in $GF(p)$, and let $a(\cdot) : GF(p) \to GF(p)$ denote the polynomial corresponding to those coefficients. The degree of $a(\cdot)$ is at most $D = \frac{\ell' e}{\log p}$. Choosing $x \in GFp$ at random means that any two distinct strings a, a' will satsify $a(x) = a'(x)$ with probability at most $D/p < \ell' e/p$. Now there are $O(1/\epsilon)$ strings which we want to distinguish (Lemma 4.3), and hence $O(\frac{1}{\epsilon^2})$ pairs. Thus, by the union bound the probability that a random point x fails to distinguish some pair is at most $O(\ell' e/(p\epsilon^2))$. To ensure that there is an x distinguishing all pairs, we choose p so as to make this expression less than one. Thus, by appropriately choosing p, we get that there exists some value x such that all the possible strings v' have different values of $v'(x)$. The needed advice is only $x, v(x)$, which requires $2 \log p = O(\log \ell' + e)$ bits. The running time of this procedure is roughly D field operations in $GF(p)$, which takes no more than $\frac{\ell' e}{\log p} \tilde{O}(\log p) = \tilde{O}(\ell' e)$ steps. (This is less than the computation time necessary in previous phases.) $\square$

The proof above uses the following technical lemma:

Lemma 4.3 *When $q \geq 10/\epsilon^2$, there are at most $O(1/\epsilon)$ candidate vectors $v' \in F^{\ell'}$ which have agreement $\epsilon/2$ with any fixed function $\tilde{f} : F^{\ell'} \times \{0,1\}^R \to F$.*

Proof. Any two distinct linear functions over F agree on at most a $1/q = \epsilon^2/10$ fraction of the points in the set $F^{\ell'} \times \{0,1\}^R$. By the Johnson bound [11, 10], any code with minimum relative distance $1 - \epsilon^2/10$ has list size $t(\epsilon) \leq 3/\epsilon$. $\square$

Finally, we can prove Theorem 4.1:

Proof (Proof of Theorem 4.1). Fix some soundness target ϵ, and choose q to be the smallest power of two greater than $10/\epsilon^2$. Let $\log(1/\epsilon) = \ell^{\gamma'}$, so that $\ell \log \frac{1}{\epsilon} = O(\ell^{1+\gamma'})$ and ϵ is negligible in ℓ.

Let $\ell' = \ell/\log q$. We will use a truth table for g to specify the bits of the matrix $A \in GF(q)^{\ell' \times \ell'}$ describing the (linear) function f to be used for the proof system. We obtain the truth table by listing the value of g on all strings of length $\log(\ell^2)$. Since $g \in E$, we can compute the truth table in time $poly(\ell)$. Moreover, since f is also linear of $GF(2)$, the protocol will be complete (based on the existence of XOR-malleable cryptosystems).

By choosing a sufficiently small constant α such that $k = \ell^\alpha$, we can ensure that the reduction from prover to proof auditor loses no more than $\ell \cdot k^{c_1} \leq \ell^{1+\gamma'}$ (additively) in both running time and required advice (see Theorem 2.1). Thus a cheating prover which uses time and advice $O(\ell^{1+\gamma'})$, no non-determinism, and success probability $\epsilon+2\cdot 4^{-k}$, can be converted to a *single-valued* i.o. proof auditor which has time and advice bounds $\ell^{1+\gamma'}$ and success rate ϵ (the single-valued property comes from the specifics of the DS reduction).

Now a proof auditor for a linear map $x \mapsto Ax$ is, in particular, a proof auditor (with at least the same success rate) for the linear function given by any row v of A. By Theorem 4.2, we can construct a verifier for an MAM proof system which computes the row vector v, and whose circuit size is $\tilde{O}(T+a+\ell \log \frac{1}{\epsilon}) = O(\ell^{1+2\gamma})$. We can modify this circuit to take an additional input $i \in \{1, ..., \ell'\}$, which tells it which row of A it should be computing, so that essentially the same reduction produces a verifier circuit of size $O(\ell^{1+2\gamma})$ which can be used verify the correctness of any bit of the matrix A[8].This contradicts the (worst-case) hardness assumption for g, and hence we get that the protocol is secure against provers of time and advice bound $\ell^{1+\gamma'}$. □

5 Assuming Complete Malleability

If we are willing to assume the existence of a completely malleable cryptosystem we are no longer forced to work with functions f which are linear. To guarantee the security of the protocol in this setup we only require that f does not have a proof auditor which is simultaneously time bounded and advice bounded. We have no candidate for arbitrarily malleable cryptosystem. Nonetheless, in this section we give two additional illustrations of the power of such a (hypothetical) cryptosystem.

5.1 Advice-Bounded Provers and Reed-Solomon Codes

Theorem 3.1 allows us to use almost any good list-decodable code, regardless of linearity. (Polynomial-time computability of any particular component of a codeword is still necessary for completeness.) A natural candidate is the Reed-Solomon code. Suppose that we want a power p gap between the advice needed by the honest prover during the proving time and the advice necessary in order to cheat with probabilitty (roughly) ϵ. If we consider polynomials of degree $d = \ell^{p-1}$ over the field $F = GF(2^\ell)$, then we get a class of functions such that (a) any function is describable by ℓ^p bits, and (b) any two distinct functions from the class agree on at most a fraction $d/2^\ell$ of the input values in F. By a standard analysis (essentially the same as in the proof of Lemma 4.3), the corresponding Reed-Solomon code has list size $t(\alpha) \leq 3/\alpha$ for any $\alpha > 4\sqrt{d/2^\ell}$.

Setting $\log \frac{1}{\epsilon} = \ell/5$ for concreteness, we can apply Theorem 3.1. Using $\alpha = \epsilon^2/2$, we see that as long as cheating provers have less than $\ell^p - \log t(\alpha) - \log(1/\epsilon) = \ell^p(1 - o(1))$ bits of advice, then there exists a function f (given by

[8] The only difficulty here is that there were $\log \ell' \log \frac{1}{\epsilon}$ bits of advice which were specific to v and hence to the index i. However, including all ℓ' possible advice strings that Arthur might change the circuit size by at most $O(\ell' \log \ell' \log \frac{1}{\epsilon})$. This is dominated by other terms in the circuit size.

some codeword) for which a cheater has at most a probability of $\epsilon + 2 \cdot 4^{-k}$ chance of breaking the protocol, whereas the honest prover requires advice $\ell \cdot k^c$ for some constant c. This in fact also requires $d/2^\ell < \epsilon^4/32$, but this holds whenever $32\ell^{p-1} < 2^{\ell/5}$, i.e. for all sufficently large ℓ.

5.2 Simultaneously Time- and Advice-Bounded Provers

To guarantee the security of the protocol in this setup we only require that f does not have a proof auditor which is simultaneously time bounded and advice bounded.

In this section we show that such a proof auditor gives rise to variants of nondeterministic circuits which compute f on a non-negligible fraction of the inputs. We can use "hardness amplification" techniques to construct (non-linear) functions f which are hard on average from functions h which are hard on the worst case. This allows us to base the security of the Dwork-Stockmeyer protocol on more standard complexity assumptions in which the time it takes to compute the hard function is an arbitrary polynomial in its hardness: There are functions computable in E which cannot be computed *on the worst case* by small Σ_3-circuits.[9] We remark that analogous assumptions are used in derandomization to obtain that $AM = NP$ [13, 17, 19] and to construct extractors for samplable distributions [24].

Theorem 5.1 *Suppose there exists a constant γ and a function $h = \{h_s\}$, $h_s : \{0,1\}^s \to \{0,1\}$ computable in time $2^{O(s)}$ such that h cannot be computed by Σ_3-circuits of size $2^{\gamma s}$, and assume the existence of a completely malleable cryptosystem. Then let n denote the length of the statement τ, and $k > n$ denote the security parameter. For every constant $p > 1$ the DS protocol is secure with soundness $\epsilon^*(k) = \Omega(2^{-k})$, dishonest prover bounds: $T^*(k) = a^*(k) = k^p$ and honest prover bounds $T(k) = a(k) = k^{O(1)}$ for some fixed constant which does not depend on p.*

As the prover is both time bounded and advice bounded we can assume that all the parameters of the single valued proof auditor are bounded by some bound S. More precisely, that $a + R + N + T < S$ where these parameters are taken from Definition 2.1. We call such an auditor *S-bounded*. We can also assume that the proof auditor isn't randomized, that is $R = 0$. This is because the auditor can get the "best" random string r as additional short advice[10]. The auditor is now a circuit $\mathcal{A}(x, w)$ of size S such that:

$$\Pr_{x \in \{0,1\}^\ell} \Big[\big(\exists z \in \{0,1\}^N (\mathcal{A}(x, z) = (1, y)) \iff y = f(x)\big) \Big] \geq \epsilon(\ell)$$

[9] A Σ_i-circuit is a circuit which can have gates which compute a complete language in Σ_i (the i'th level of the polynomial hierarchy).

[10] This was *not* possible in previous proofs in this paper. For example, in the advice bounded case R could be much larger than a, making it impossible to store the "best" random tape.

In words, on ϵ fraction of the inputs x, there is a unique answer y such that every "nondeterministic guess" z on which $\mathcal{A}$ answers is labelled with y. We have no guarantee on how $\mathcal{A}$ behaves on the remaining x's. In particular it may be the case that for every z, the first output of $A(x,z)$ is 0, or that there are contradictory answers (different z's lead to different y's such that $\mathcal{A}(x,z) = (1,y)$).

We first observe that we can transform $\mathcal{A}$ into a circuit C (with an NP oracle) such that C *does* have a unique value for every input.

Lemma 5.2 *There is a circuit C with NP-oracle of size $S^{O(1)}$ such that*

$$\Pr_{x \in \{0,1\}^{\ell}}[C(x) = f(x)] \geq \epsilon(l)$$

Proof. Let $\mathcal{A}_1$ (resp. $\mathcal{A}_2$) denote the first (resp. second) output of $\mathcal{A}$. On input x, C uses its NP-oracle to check if $x \in \{x | \forall z. \mathcal{A}_1(x,z) = 0\}$. In that case x is not one of the good inputs on which $\mathcal{A}$ agrees with f and C answers arbitrarily. If x is good then C uses its NP-oracle to find z such that $\mathcal{A}(x,z) = (1,y)$ and outputs y. $\square$

We can now use a result by Trevisan and Vadhan [24] (see also [22]) which shows that if f is a low-degree multivariate polynomial then C can be transformed into a small circuit C' (with Σ_3-oracle) which computes f correctly on every input.

Theorem 5.3 *[24] Let F be a finite field (with some fixed, efficient representation), and let $f : F^t \to F$ be a polynomial of total degree at most d. If there is a Σ_i-circuit C which computes f correctly on at least an $\epsilon = c(\sqrt{d/|F|})$ fraction of points (for some constant c) then there is a Σ_{i+2}-circuit C' with size $poly(|C|, d)$ which computes f correctly everywhere.*

A nice feature of this result is that the size of C' does not depend on ϵ. This will allow us to use very small ϵ which translates into negligible success probability of the cheating prover. We now recall that any function can be extended into a low degree polynomial.

Definition 5.1 (Low-degree extension). *The low degree extension of a function $h : \{0,1\}^s \to \{0,1\}$ into a multivariate polynomial $f : F^t \to F$ over a field F with at leats $2^{s/t}$ elements works by taking some subset $H \subseteq F$ of size $2^{s/t}$ and identifying H^t with $\{0,1\}^s$. For every $x \in H^t$ we define $f(x) = h(x)$. We can now interpolate and extend f into a polynomial in t variables with degree at most $|H|$ in every variable. The total degree of such a polynomial is at most $d = |H|t = 2^{s/t}t$.*

It immediately follows that:

- If h is computable in time $2^{O(s)}$ then f is also computable in time $poly(2^s, \log |F|)$.
- A circuit which computes f induces a circuit which computes h.

Lemma 5.4 *For every constant $\gamma > 0$ there exists constant $\gamma' > 0$ such that if there exists a function $h = \{h_s\}$, $h_s : \{0,1\}^s \to \{0,1\}$ computable in time $2^{O(s)}$ such that h cannot be computed by Σ_3-circuits of size $2^{\gamma s}$, then for every $2 < a \leq 2^s$ there is a function $f = \{f_s\}$, $f_s : \{0,1\}^{as} \to \{0,1\}^{as}$ such that f is computable in time $2^{O(s)}$ and for every s and every NP-circuit C of size $2^{\gamma' s}$:*

$$Pr_{x \in \{0,1\}^{as}}[C(x) = f_s(x)] < 2^{-\Omega(as)}$$

Proof. We let f_s be the low-degree extension of h_s, taking $t = c's/\gamma$ (where c' is a constant to be determined later), and $|F| = 2^{as/t}$. We note that f is computable in time $poly(2^s, \log |F|) = 2^{O(s)}$. By Theorem 5.3 any NP-circuit C of size $2^{\gamma' s}$ which computes f correctly on $\epsilon = c\sqrt{(2^{s/t}t)/2^{as/t}} < 2^{-\Omega(as)}$ can be transformed into a Σ_3-circuit C' of size $poly(2^{\gamma' s}, 2^{s/t}t)$ which computes f everywhere. We choose γ' small enough and c' large enough so that the size of C' is at most $2^{\gamma s}$. □

We conclude that the assumption of Lemma 5.4 is sufficient for the security of the Dwork-Stockmeyer protocol.

Proof. (of Theorem 5.1) On inputs of length n and security parameter $k > n$ we choose $s = c \log k$ for some constant $c > 1$ to be determined later. We use Lemma 5.4 with $a = k$ We obtain a function f_s that takes inputs of length $\ell = as = O(ck \log k)$, is computable in time poly(k) and is hard for NP-circuits of size $2^{\gamma' s} = k^{c\gamma'}$. By Lemma 5.2, f_s is hard for ${}^{(c\gamma')/c'}$-bounded proof auditors where c' is the constant hidden in the $O(\cdot)$ notation in Lemma 5.2. By Theorem 2.1 the DS-protocol is $(2^{-\Omega(as)} + 2 \cdot 4^{-k})$-sound against provers with $T^* = T - \Omega(\ell k^{O(1)})$. It follows that for every constant p we can choose the constant c so that $T^* > k^p$ and $\epsilon^*(k) < O(4^{-k})$. Note that the honest prover runs in time $\ell k^{O(1)} = k^{O(1)}$ for some fixed constant that doesn't depend on c. □

References

1. Y. Aumann, Y. Z. Ding, and M. Rabin, Everlasting Security in the Bounded Storage Model. *IEEE Transactions on Information Theory*, April, 2002.
2. B. Barak, S. J. Ong, S. Vadhan. Derandomization in Cryptography In *CRYPTO* 2003.
3. G. Brassard, D. Chaum, C. Crépeau. Minimum Disclosure Proofs of Knowledge. *J. Comput. Sys. Sci.*, 37(2): 156-189 (1988).
4. B. Chor and O. Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM Journal on Computing*, 17(2):230–261, April 1988.
5. C. Dwork and M. Naor, Zaps and their applications, *Proc. 41st IEEE Symp. on Foundations of Computer Science*, 2000, pp. 283–293.
6. C. Dwork and L. Stockmeyer. 2-Round Zero Knowledge and Proof Auditors. *Proc. 34th ACM Symp. on Theory of Computing*, 2002.
7. O. Goldreich and L. Levin, A hard-core predicate to any one-way function, *Proc. 21 ACM Symp. on Theory of Computing*, , 1989.

8. S. Goldwasser and S. Micali, Probabilistic encryption, *J. Comput. Syst. Sci.* 28 (1984), pp. 270–299.
9. S. Goldwasser, S. Micali, and C. Rackoff, The knowledge complexity of interactive proof systems, *SIAM J. Comput.* 18(1) (1989), pp. 186–208.
10. V. Guruswami, M. Sudan. Extensions to the Johnson bound. Manuscript, 2001.
11. S. Johnson. A new upper bound for error-correcting codes. *IEEE Trans. on Info. Theory*, 9, 1963, pp. 198–205.
12. J. Kamp and D. Zuckerman. Deterministic Extractors for Bit-Fixing Sources and Exposure-Resilient Cryptography. *Proc. IEEE Symp. on Foundations of Computer Science*, 2003.
13. A. R. Klivans and D. van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. In *31st Annual ACM Symposium on Theory of Computing*, pages 659–667, 1999.
14. C. Lu. Hyper-encryption against Space-Bounded Adversaries from On-Line Strong Extractors. In *CRYPTO* 2002, pp. 257–271.
15. Y. Mansour, N. Nisan and P. Tiwari. The computational complexity of universal hashing. In *22nd Annual ACM Symposium on Theory of Computing*, 1990.
16. U. Maurer. Conditionally-Perfect Secrecy and a Provably-Secure Randomized Cipher. *J. Crypto.*, 5(1), pp. 53–66, 1992.
17. P. B. Miltersen and N. V. Vinodchandran. Derandomizing Arthur-Merlin games using hitting sets. In *40th Annual Symposium on Foundations of Computer Science (FOCS '99)*, pp. 71–80, 1999.
18. R. Rivest, L. Adleman and M. Dertouzos. On data banks and privacy homomorphisms. In *Foundations of Secure Computation*, R. de Millo et al, eds, 1978.
19. R. Shaltiel and C. Umans. Simple extractors for all min-entropies and a new pseudo-random generator. In *42nd IEEE Symposium on Foundations of Computer Science (FOCS-01)*, pages 648–657, 2001.
20. Amir Shpilka, Personal communication, October 2002.
21. L. Stockmeyer. On approximation algorithms for $\#P$. *SIAM J. Comput.* 14(4) (1985), pp. 849–861.
22. M. Sudan, L. Trevisan, and S. Vadhan. Pseudorandom generators without the XOR lemma. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*, 1999.
23. L. Trevisan. Construction of extractors using pseudorandom generators. In *Proceedings of the 31st ACM Symposium on Theory of Computing*, 1999.
24. L. Trevisan and S. Vadhan. Extracting randomness from samplable distributions. In *41st Annual Symposium on Foundations of Computer Science*, pp. 32–42.
25. S. Vadhan. On Constructing Locally Computable Extractors and Cryptosystems in the Bounded Storage Model. In *CRYPTO* 2003.