Pseudo-Random Generators for All Hardnesses

Christopher Umans [1]

Computer Science Department, Caltech, Pasadena CA 91125

Abstract

Given a function $f : \{0,1\}^{\log n} \to \{0,1\}$ with circuit complexity s, we construct a pseudo-random generator that stretches a random seed of length $O(\log n)$ into a string of $m = s^{\Omega(1)}$ pseudo-random bits that fool circuits of size m. The construction works for *any* hardness s, giving an optimal hardness vs. randomness tradeoff with a direct and self-contained proof. A key element in our construction is an augmentation of the standard low-degree extension encoding that exploits the field structure of the underlying space in a new way.

Key words: pseudo-random number generators, derandomization, complexity classes, low-degree extension

1 Introduction

One of the most intriguing and active areas in complexity theory concerns the power of randomized algorithms. It is now known that BPP (randomized polynomial-time with two-sided error) and RP (randomized polynomial-time with one-sided error) coincide with P *under the plausible assumption* that some language in E requires exponential-size circuits. [2] In other words, within the context of polynomial running time, randomness probably does not help. This remarkable conclusion is the result of a long line of research, initiated by Blum, Micali and Yao [5,26], focused on "hardness vs. randomness" tradeoffs [19,3,9], which culminated in a result of Impagliazzo and Wigderson [13] stating that BPP = P if some language in E requires exponential-size circuits. Subsequently this result has been extended and simplified by several authors

Email address: `umans@cs.caltech.edu` (Christopher Umans).

[1] Much of this work was done while the author was a postdoc at Microsoft Research.

[2] Recall that $\mathrm{E} = \mathrm{DTIME}\left(2^{O(n)}\right)$.

[22,11,12,20]. In general, given a hard language L in E, these tradeoffs give deterministic simulations of BPP whose running time depends inversely on the circuit complexity of L.

The rough outlines of the derandomization picture – namely, the hardness assumption under which BPP = P – have been known for several years now. Only recently have the finer details been filled in giving the "correct" quantitative tradeoff, up to a polynomial slowdown, between hardness and randomness, stated in the following theorem of Shaltiel and Umans:

Theorem 1 ([20]) *If there exists a function family $f = \{f_n\}$ in E with circuit complexity at least $s(n)$, then*

$$BPTIME(\ell) \subseteq DTIME(2^{O(s^{-1}(\ell^{O(1)}))}),$$

where ℓ is the running-time of the probabilistic procedure.

There are currently two natural routes to proving Theorem 1. The less direct route comprises two steps: first, one constructs a *hitting-set generator* (HSG) that converts s bits of hardness into $s^{\Omega(1)}$ bits of 1-*sided* pseudo-randomness. This was the route taken by Shaltiel and Umans; we give one necessary definition and the result below:

Definition 2 *(ϵ-HSG) An* ϵ-HSG for size s *is a function* $H : \{0,1\}^t \to \{0,1\}^m$ *such that for all circuits C of size at most s whose acceptance probability is at least ϵ, there exists a z such that $C(H(z)) = 1$.*

Theorem 3 ([20]) *Given a function $f : \{0,1\}^{\log n} \to \{0,1\}$ with circuit complexity at least s, one can construct a $1/m$-HSG $H : \{0,1\}^{O(\log n)} \to \{0,1\}^m$ for size m, with $m = s^{\Omega(1)}$; moreover, H can be computed in $n^{O(1)}$ time given access to the truth table of f.*

The second step utilizes a result of Andreev, Clementi, and Rolim [1] (see [6,2] for a simpler proof) stating that HSGs suffice to derandomize BPP; Theorem 1 then follows.

An alternative, more direct route entails building a *pseudo-random generator* (PRG) that converts s bits of hardness into $s^{\Omega(1)}$ bits of *2-sided* pseudo-randomness. Motivated in part by the connection between extractors and PRGs discovered by Trevisan [24], considerable attention has recently been focused on attaining such a construction [22,11,12], so far without success. The best known construction prior to this paper is due to Impagliazzo, Shaltiel, and Wigderson [12]; we give one necessary definition and that result below:

Definition 4 *(ϵ-PRG) An* ϵ-PRG for size s *is a function* $G : \{0,1\}^t \to$

$\{0,1\}^m$ *such that for all circuits C of size at most s:*

$$|\Pr_z[C(G(z))=1] - \Pr_x[C(x)=1]| \le \epsilon. \quad (1)$$

Theorem 5 ([12]) *Given a function $f : \{0,1\}^{\log n} \to \{0,1\}$ with circuit complexity at least s, one can construct an $1/m$-PRG $G : \{0,1\}^{O(\log n)} \to \{0,1\}^m$ for size m with $m = s^{\Omega(1/\log\log\log n)}$; moreover, G can be computed in $n^{O(1)}$ time given access to the truth table of f.*

Theorem 5 is *not* sufficient to obtain Theorem 1. Our main result is the construction of an "optimal" PRG that converts s bits of hardness into $s^{\Omega(1)}$ bits of *2-sided* pseudo-randomness, improving Theorem 5 and resolving an open question raised by Impagliazzo et al. [12] and Shaltiel and Umans [20].

Theorem 6 (this paper) *Given a function $f : \{0,1\}^{\log n} \to \{0,1\}$ with circuit complexity at least s, one can construct an $1/m$-PRG $G : \{0,1\}^{O(\log n)} \to \{0,1\}^m$ for size m with $m = s^{\Omega(1)}$; moreover, G can be computed in $n^{O(1)}$ time given access to the truth table of f.*

This theorem gives a direct, self-contained proof of Theorem 1, without appealing to the Andreev et al. result [1]. Our PRG also avoids the explicit hardness amplification phase that was required in most prior (Nisan-Wigderson-style) PRGs, by converting worst-case hardness *directly* into 2-sided randomness. The only other such PRG construction which circumvents hardness amplification [20] has a non-optimal seed length of $O(\log^2 n/\log s)$ and hence cannot be used to derive Theorem 1. Altogether, our construction gives a self-contained and relatively simple proof of the optimal tradeoffs between (non-uniform) hardness and randomness for the full range of parameters, and in particular subsumes Impagliazzo and Wigderson [13] and subsequent improvements [11,12,20].

Theorem 6 also shows in a precise sense the *equivalence* (up to a polynomial) between hardness and 2-sided pseudo-randomness, something that was not known previously. It is folklore (see also Impagliazzo et al. [11]) that s-bits of 2-sided pseudo-randomness, in the form of a $\frac{1}{2}$-PRG for size s with seed-length t, gives rise to s bits of hardness, in the form of a function $f : \{0,1\}^{t+1} \to \{0,1\}$ with circuit complexity at least s. The other direction is considerably more difficult; our main theorem gives the first conversion in the other direction with only polynomial losses: from the aforementioned function, we construct a $\frac{1}{2}$-PRG for size $s^{\Omega(1)}$ with seed length $O(t)$. Thus we give the first *optimal* reduction from hard functions to PRGs, in the sense of Impagliazzo et al. [11] – any reduction that does significantly better produces a harder function than the one originally supplied to it. Another reason the parameters cannot be significantly improved is that such a reduction produces extractors, as discovered by Trevisan [24], and so is subject to the information-theoretic limitations on extractor parameters [11].

From another persepective, PRGs, like extractors, have been used to prove statements beyond their intended derandomization consequence (see, e.g., [15,14,17,10]). From this viewpoint, PRGs are "tools" for converting between two fundamental resources – hardness and pseudo-randomness – and it is useful to have optimal PRGs at our disposal for use in complexity theoretic and other applications.

Finally, our PRG builds on the new algebraic PRG construction of Shaltiel and Umans [20] ("the SU PRG"), and it is possible that some of the algebraic structure that we exploit, which goes beyond that which was used previously, may be helpful in improving the extractor constructions in that paper, which use the same framework. Indeed, the PRG construction in this paper yields extractors via the connection between extractors and certain PRGs noticed by Trevisan [24]. It remains an open problem to construct optimal extractors, and the new techniques of Ta-Shma, Zuckerman, and Safra [23] and Shaltiel and Umans [20] may be able to be pushed further with additional insight into the algebraic structure of these constructions; we hope some of the ideas in this paper can be useful to this end.

The remainder of the paper is organized as follows. In Section 2 we outline the ideas behind the SU PRG construction, describe the apparent barrier to obtaining optimal PRGs, and give an informal description of the new ideas we use to overcome this barrier. In Section 4 we present the "augmented low-degree encoding" which is perhaps the main conceptual contribution of this work. Finally in Sections 5 and 6 we give the construction of the new PRG and a proof of the main theorem, respectively.

2 Overview of our results

In addition to the HSG in Theorem 2, Shaltiel and Umans also construct a PRG [20], but it only has the desired seed length when it is constructed from an exponentially hard function. Nevertheless, it is significant in that it is the first PRG construction not based on the Nisan-Wigderson construction [19], with the exception of the Blum-Micali-Yao-style PRGs [5,26], which are based on a qualitatively stronger hardness assumption. We are able to make the this construction work for all hardnesses, using some new ideas which we outline below. Since the starting point for our construction is the SU PRG, we outline that construction before explaining our improvements.

2.1 The SU PRG construction [20]

As a first step we take the truth table of the hard function (which is just a n-bit string x), and encode it in a multivariate low-degree polynomial p_x so that given i, we can easily compute a location at which the evaluation of the polynomial gives us x_i. This is almost the usual "low-degree extension" encoding of x, however for technical reasons it must be slightly non-standard [20]. The polynomial p is defined over the vector space $\mathbb{F}_q^d$ (where $\mathbb{F}_q$ denotes the finite field of q elements), but we can also view $\mathbb{F}_q^d$ as the *extension field* $\mathbb{F}_{q^d}$. The multiplicative group of $\mathbb{F}_{q^d}$ is cyclic, so there is an $\alpha \in \mathbb{F}_{q^d}$ (a *primitive element*) that generates this group. Once we have α, we can write down the evaluations of p_x at every point in the order imposed by α – namely $p_x(\alpha^0), p_x(\alpha^1), p_x(\alpha^2), \ldots$.

A first attempt at building a PRG is to use the seed to choose a location in this list of evaluations of p_x, and output m successive evaluations in the list [3] – in other words, define $G_x(i) = p_x(\alpha^i) \circ p_x(\alpha^{i+1}) \circ \cdots \circ p_x(\alpha^{i+m-1})$. It turns out that this is an extractor [20] but not yet a PRG, and to see why we need to examine the proof technique used by Shaltiel and Umans [20] and see where it breaks. The first step of the proof is standard: if this construction is *not* a PRG, then there is an efficient *next-element predictor* g that given the first $m-1$ output elements, outputs the m-th element with non-negligible probability. In our case, the next-element predictor predicts $p_x(\alpha^{j+m})$ from the $m-1$ previous evaluations $p_x(\alpha^{j+1}), p_x(\alpha^{j+2}), \ldots, p_x(\alpha^{j+m-1})$, for any j.

Now, we want to use g to build a small circuit that computes x_i given input i, contradicting the hardness of x. The idea (first used by Ta-Shma et al. [23]) is roughly this: we hardwire the first $m-1$ evaluations in the list into the circuit, and use g to predict the m-th evaluation; we then use g on $m-2$ evaluations in the list together with the just-predicted value to predict the $(m+1)$-st evaluation, and so on down the list. If g were *errorless*, this would give us a way to compute *every* evaluation, and since x_i is one of these evaluations, we would be able to compute x_i given i. In actuality, g makes many errors, but we can overcome that by exploiting the error-correcting properties of low-degree polynomials. Specifically, we use the predictor along a *random curve* at each step, and apply error correction along that curve. A good portion of [20] was devoted to making this general idea actually work, and we won't delve into the details here. The conclusion is that using these ideas it is possible to simulate an errorless predictor given access to one that makes errors.

But, there is a more serious problem with our PRG attempt. The running

[3] In this discussion we ignore the distinction between outputting m field elements and outputting m bits. There is little harm in this simplification: Lemma 13 describes how to convert "q-ary" PRGs (Definition 11) into ordinary ones.

time of the above procedure is proportional to the number of steps it takes. No matter how we "embed" x into p_x, there will be some i for which x_i is at least n steps away, and the size of our circuit must therefore be at least n, which is far too large to contradict the hardness of x, which is *a priori* less than n. In fact it seems that any such sequential use of a single predictor will suffer from the same problem.

Shaltiel and Umans overcome this problem by arranging to have *several* different predictors available, working at larger and larger "strides." By switching from one predictor to the next, they are able to "travel" from the locations of the hardwired evaluations to any given location in $\mathbb{F}_q^d$ in only poly(m) steps. The price for having several predictors is that they end up with several "candidate" PRGs, only one of which is guaranteed to be a PRG. The standard thing to do in this situation is to XOR the candidate PRGs with independent seeds, which produces a single PRG, but blows up the seed length. It is this blow-up that prevents the SU PRG from achieving logarithmic seed length for all except exponential hardnesses.

2.2 Our improvements

In this paper we suggest a method for making a *single* predictor work, by altering the *encoding* of x. The general idea is to force the evaluation of p_x at any given location in $\mathbb{F}_q^d$ to be *repeated* in a small number of carefully chosen different locations in $\mathbb{F}_q^d$. Then, after using the predictor for a small number of steps, we will have learned the evaluations of p_x at enough "faraway" locations to "jump" to a remote location and begin predicting there.

The challenge is to arrange the locations of repeated values so that this "jumping" allows us to get to any location in $\mathbb{F}_q^d$ in only poly(m) steps (and that the required sequence of predictor invocations can be easily computed). It turns out that the following scheme works: we ensure that $p_x(\alpha^i) = p_x(\alpha^{iq})$ for all i – here we use critically that $\alpha^{q^d} = \alpha$, so each location has at most $d-1$ associated locations with the same value. We illustrate a single "jump" in this scheme. Suppose for some j we have used the predictor to learn values $p_x(\alpha^j), p_x(\alpha^{j+1}), \ldots, p_x(\alpha^{j+(m-1)q})$. Then we also know the values

$$p_x(\alpha^{jq^{d-1}}), p_x(\alpha^{jq^{d-1}+1}), \ldots, p_x(\alpha^{jq^{d-1}+(m-1)})$$

since multiplying the exponent on α by q in the latter sequence of evaluations gives exponents $j, j+q, j+2q, \ldots j+(m-1)q$. We can therefore "jump" to location $\alpha^{jq^{d-1}}$, and begin predicting there, since we know the requisite $m-1$ values needed to get the predictor started. In the actual encoding, we do not quite obtain equality between $p_x(\alpha^i)$ and $p_x(\alpha^{iq})$ – but each can be easily computed from the other, which suffices for our application. In this simplified presentation we are also ignoring the fact that all of the techniques of Shaltiel and Umans [20] for handling error-correction along curves, etc... must be integrated, but with some effort this can be done.

In order to handle error-correction, we need p_x to be a low-degree polynomial, but then it is impossible to *simultaneously* enforce the constraint that $p_x(\alpha^i) = p_x(\alpha^{iq})$ for *all* i. We finesse this issue by allowing p_x to map into an expanded range $\mathbb{F}_{q^d}$ (instead of $\mathbb{F}_q$). The encoding is no longer a low-degree polynomial; however, viewing $\mathbb{F}_{q^d}$ as $\mathbb{F}_q^d$, we ensure that *each coordinate* of p_x is a low-degree polynomial. This allows us to continue to use the error-correction properties of the original encoding while also benefiting from the ability to "jump" between locations using a single predictor.

It is interesting to note that our PRG amounts to combining the candidate (q-ary) PRGs of Shaltiel and Umans [20], using *the same seed* for all candidates instead of independent seeds. Of course this method of combining candidate generators cannot work in general, and so it is not surprising that we rely heavily on specific (algebraic) features of the construction.

The encoding we end up with turns out to be quite nice mathematically. Suppose we write our original list of evaluations of p_x in a row and write below it the $d-1$ permutations of the same list obtained by applying each of the non-trivial field automorphisms $\sigma \in G$, where G is the Galois group of $\mathbb{F}_{q^d}$ over $\mathbb{F}_q$. Then the i-th element of the new encoding is the i-th column in this array. Just as the SU PRG used critically that multiplication by α corresponds to a *linear* transform of $\mathbb{F}_q^d$, we use, in addition, that all $\sigma \in G$ are linear transforms of $\mathbb{F}_q^d$. Finally, we remark that the main new ingredient of our encoding – the repeated values at different locations – has a clean interpretation in terms of the Fourier Transform of the encoding, which we point out in the appendix (Section A).

3 Finite field preliminaries

In this section we gather a number of facts regarding finite fields that we will use in the remainder of the paper. A more detailed presentation of this material (with the exception of Lemma 7) can be found in [16], or any of several other standard algebra texts.

We will be working with the field of q elements, $\mathbb{F}_q$, and the extension field of q^d elements $\mathbb{F}_{q^d}$, which is a vector space of dimension d over $\mathbb{F}_q$. In computations, we assume that elements of $\mathbb{F}_{q^d}$ are represented as vectors in $\mathbb{F}_q^d$ with respect to some basis $\{\beta_0, \beta_1, \ldots, \beta_{d-1}\}$, where the β_i are elements of $\mathbb{F}_{q^d}$. The *standard*

$$\begin{tikzcd}[column sep=small] & \mathbb{F}_{q^d} \arrow[dl, dash] \arrow[dr, dash] & \\ \mathbb{F}_{h^d} \arrow[dr, dash] & & \mathbb{F}_q \arrow[dl, dash] \\ & \mathbb{F}_h & \end{tikzcd}$$

Fig. 1. Containment diagram for Lemma 7.

basis is $\left\{1, z, z^2, \ldots, z^{d-1}\right\}$, where z is an indeterminate, and we think of $\mathbb{F}_{q^d}$ as polynomials in z modulo an irreducible degree d polynomial $p(z)$.

A very useful alternative basis is a *normal basis* which is a basis of the form $\{\sigma\alpha | \sigma \in G\}$, where α is an element of the extension field and G is the Galois group of the extension. In our case the Galois group of $\mathbb{F}_{q^d}$ over $\mathbb{F}_q$ is cyclic of order d and generated by the *Frobenius automorphism*, $\alpha \mapsto \alpha^q$, for $\alpha \in \mathbb{F}_{q^d}$; the group operation is composition. Thus a normal basis for $\mathbb{F}_{q^d}$ over $\mathbb{F}_q$ has the form $\left\{\alpha, \alpha^q, \alpha^{q^2}, \ldots, \alpha^{q^{d-1}}\right\}$, for some $\alpha \in \mathbb{F}_{q^d}$. The "normal basis theorem," found in most algebra texts (e.g., [16]), states that a normal basis exists for any finite Galois extension of fields, which certainly covers our situation. For our purposes it will be sufficient to observe that a normal basis can be found in time polynomial in the size of the extension field, by exhaustive search.

We will also need the following lemma which allows us to construct certain extension fields using an irreducible polynomial whose coefficients all lie in a subfield of the base field.

Lemma 7 *Let h be a prime power, let c and d be relatively prime positive integers, and set $q = h^c$. Then there exists a degree d irreducible polynomial over $\mathbb{F}_q$ all of whose coefficients lie in $\mathbb{F}_h$.*

PROOF. The field $\mathbb{F}_{q^d}$ contains both $\mathbb{F}_{h^d}$ and $\mathbb{F}_q$ (refer to Figure 1). The field $\mathbb{F}_{h^d}$ is obtained by adjoining to $\mathbb{F}_h$ a root $\omega \in \mathbb{F}_{q^d}$ of a degree d polynomial $p(z)$ irreducible over $\mathbb{F}_h$. If we adjoin ω to $\mathbb{F}_q$, we obtain the field $\mathbb{F}_q(\omega)$, which contains both $\mathbb{F}_{h^d}$ and $\mathbb{F}_q$ and is contained in $\mathbb{F}_{q^d}$. We claim that no proper subfield of $\mathbb{F}_{q^d}$ can contain both $\mathbb{F}_q$ and $\mathbb{F}_{h^d}$. To see this, recall that the field $\mathbb{F}_{h^a}$ contains a subfield with $h^{a'}$ elements iff a' divides a. Therefore, if $\mathbb{F}_q(\omega)$ is a subfield of $\mathbb{F}_{q^d} = \mathbb{F}_{h^{cd}}$, then it must have $h^{a'}$ elements for some a' that divides cd. Since $\mathbb{F}_q(\omega)$ contains subfields with $q = h^c$ elements and h^d elements, we have that c and d both divide a'. As c and d are relatively prime, it must be that $a' = cd$. Therefore $\mathbb{F}_q(\omega) = \mathbb{F}_{h^{cd}} = \mathbb{F}_{q^d}$, and so $p(z)$ must be irreducible over $\mathbb{F}_q$. $\square$

A fact that we will use heavily is that the Frobenius map $\alpha \mapsto \alpha^q$ is $\mathbb{F}_q$-*linear*; that is, $(\alpha+\beta)^q = \alpha^q + \beta^q$ for all $\alpha, \beta \in \mathbb{F}_{q^d}$ and $(a\beta)^q = a\beta^q$ for all $a \in \mathbb{F}_q$ and $\beta \in \mathbb{F}_{q^d}$. The latter equality follows from the fact that $a^q = a$ if and only if $a \in \mathbb{F}_q$. One nice feature of a normal basis is that exponentiation by q is simply a cyclic shift of the coordinates.

Finally, recall that the multiplicative group of a finite field is cyclic; a generator of this group is called a *primitive element* of the field.

4 The encoding

In this section we describe a function $\widetilde{\mathrm{LDE}}_x$ that encodes a string $x \in \{0,1\}^n$. This function resembles a standard "low-degree extension" with an important additional "periodicity" property that we describe below. We begin by describing a non-standard low-degree extension used in [20] that is tailored for our purposes, but lacks the periodicity property. We then use this low-degree extension to describe the "augmented low-degree extension" that is used in the remainder of the paper.

4.1 A low-degree extension

Pick parameters d, h, and q, and let $\left\{\beta, \beta^q, \beta^{q^2}, \ldots, \beta^{q^{d-1}}\right\}$ be a normal basis for $\mathbb{F}_{q^d}$ over $\mathbb{F}_q$. From here on, we assume that elements of $\mathbb{F}_{q^d}$ are stored as vectors in $\mathbb{F}_q^d$ with respect to this basis; we also identify $\mathbb{F}_q^d$ with $\mathbb{F}_{q^d}$ via this basis. Let α be a primitive element of $\mathbb{F}_{q^d}$.

String $x \in \{0,1\}^n$ is the string we wish to encode. The usual way of obtaining a low-degree extension of x is to pick a set $H \subseteq \mathbb{F}_q$ and an efficiently-computable 1-1 embedding function $e : [n] \to H^d$, and to encode x as a d-variate low-degree polynomial p_x over $\mathbb{F}_q^d$ such that $p_x(e(i)) = x_i$ for all $i \in [n]$. The polynomial p_x can be obtained by interpolation. In applications, one is typically given i and asked to obtain $p_x(e(i))$ efficiently from some function $f : \mathbb{F}_q^d \to \mathbb{F}_q$ that has reasonably large agreement with p_x. In our setting, we are given i, and we will have the ability to obtain $p_x(\alpha^j)$ given any j, but we run into trouble in trying to exploit this: Computing the j for which $\alpha^j = e(i)$ is a Discrete Log problem, which is presumed to be very hard.

To overcome this difficulty, we take our set H to be a special set, namely, the subfield $\mathbb{F}_h$. String x is then embedded in the locations $T = \mathbb{F}_h^d \setminus \{0\}$, and by Lemma 7 we can ensure that these locations are a *subgroup* of the multiplicative group of $\mathbb{F}_{q^d}$ (more on this below). In other words,

$$T = \left\{\alpha^r, \alpha^{2r}, \alpha^{3r}, \ldots \alpha^{(h^d-1)r}\right\}.$$

Now, if we embed x_i at location α^{ri}, it becomes easy to compute the j for which $\alpha^j = e(i)$ – it is simply ri, and we know i.

We formalize these ideas below. We desire a function $\mathrm{LDE}_x : \mathbb{F}_q^d \to \mathbb{F}_q$ with the following properties:

($P1$) LDE_x is a d-variate polynomial with total-degree at most dh, and
($P2$) there exists a function $\ell : [n] \to [q^d - 1]$ such that $\mathrm{LDE}_x(\alpha^{\ell(i)}) = x_i$ for all $i \in [n]$, computable in $\mathrm{poly}(\log n, d \log q)$ time.

As discussed above, this is more restrictive that the usual scenario in which we expect a function $e : [n] \to \mathbb{F}_q^d$ for which $\mathrm{LDE}_x(e(i)) = x_i$; here we require that the function have the form $e(i) = \alpha^{\ell(i)}$. Such a low-degree extension can be constructed efficiently as long as some modest restrictions on h, d, and q are met:

Theorem 8 *Given $x \in \{0,1\}^n$, there exists a function $LDE_x : \mathbb{F}_q^d \to \mathbb{F}_q$ satisfying ($P1$) and ($P2$) provided h is a prime power, $h^d > n$, q is a power of h, and $(\log_h q, d) = 1$. Given x, LDE_x can be computed in poly(q^d) time.*

PROOF. Since q is a power of h we have $\mathbb{F}_h \subseteq \mathbb{F}_q$. As usual, we view the extension field $\mathbb{F}_{q^d}$ as $\mathbb{F}_q[z]$ modulo a degree d polynomial $p(z)$ that is irreducible over $\mathbb{F}_q$; by Lemma 7, we may assume that $p(z)$ has coefficients in $\mathbb{F}_h$. This implies that the set

$$T = \left\{\sum_{i=0}^{d-1} a_i z^i | a_i \in \mathbb{F}_h\right\} \setminus \{0\}$$

is a subgroup of the multiplicative group of $\mathbb{F}_{q^d}$; it has order $h^d - 1$ and so setting $r = (q^d-1)/(h^d-1)$, we have $T = \{\alpha^{rj}\}_{j=1,2,\ldots,h^d-1}$. Note that we are using the fact that the multiplicative group is cyclic and therefore there is a *unique* subgroup of order $h^d - 1$.

Now, for $i = 0, 1, \ldots, n-1$, we compute vector representations for the elements of T, i.e., we compute a's in $\mathbb{F}_h$ so that $\alpha^{ri} = \sum_{j=0}^{d-1} a_j^{(i)} z^j$. The following standard low-degree extension $\mathrm{LDE}'_x : \mathbb{F}_q^d \to \mathbb{F}_q$ is not yet our final polynomial:

$$\mathrm{LDE}'_x(y_0, y_1, \ldots y_{d-1}) = \sum_{i=0}^{n-1} x_i \cdot \prod_{j=0}^{d-1} \frac{\prod_{b\in\mathbb{F}_h, b\neq a_j^{(i)}}(y_j - b)}{\prod_{b\in\mathbb{F}_h, b\neq a_j^{(i)}}(a_j^{(i)} - b)}. \tag{2}$$

It is clear that LDE'_x has total degree at most hd. Now, let $L : \mathbb{F}_q^d \to \mathbb{F}_q^d$ be the linear transform corresponding to a change of basis from $\left\{\beta, \beta^q, \ldots, \beta^{q^{d-1}}\right\}$ to $\left\{1, z, \ldots, z^{d-1}\right\}$. Our final polynomial LDE_x is defined as

$$\mathrm{LDE}_x(y_0, \ldots, y_{d-1}) = \mathrm{LDE}'_x(L(y_0, \ldots, y_{d-1})). \tag{3}$$

Clearly LDE_x has total degree at most hd and therefore satisfies property $(P1)$. Defining $\ell(i) = ri$ we see that:

$$\begin{aligned}\mathrm{LDE}_x(\alpha^{\ell(i)}) &= \mathrm{LDE}_x(\alpha^{ri}) = \mathrm{LDE}'_x(L(\alpha^{ri})) \\ &= \mathrm{LDE}'_x(a_0^{(i)}, a_1^{(i)}, \ldots, a_{d-1}^{(i)}) = x_i,\end{aligned}$$

and ℓ is clearly easy to compute, so property $(P2)$ is satisfied.

Finally, we note that the irreducible polynomial $p(z)$ can be found by exhaustive search in time $\mathrm{poly}(q^d)$, as can the primitive element α, and the normal basis. Thus LDE_x can be computed in $\mathrm{poly}(q^d)$ time. $\square$

4.2 The augmented low-degree extension

We are now ready to describe the encoding we need for our PRG construction. Let $\sigma : \mathbb{F}_{q^d} \to \mathbb{F}_{q^d}$ be the Frobenius map and let $\pi_i : \mathbb{F}_q^d \to \mathbb{F}_q$ be the projection of the i-th coordinate, for $i = 0, 1, \ldots, d-1$. We are seeking a function $\widetilde{\mathrm{LDE}}_x : \mathbb{F}_{q^d} \to \mathbb{F}_{q^d}$ with the following properties:

$(\tilde{P}1)$ for all i, $\pi_i\widetilde{\mathrm{LDE}}_x$ is a d-variate polynomial with total degree at most dh, and
$(\tilde{P}2)$ there exists a function $\ell : [n] \to [q^d - 1]$ such that $\pi_0(\widetilde{\mathrm{LDE}}_x(\alpha^{\ell(i)})) = x_i$ for all $i \in [n]$, computable in time $\mathrm{poly}(\log n, d \log q)$, and
$(\tilde{P}3)$ $\sigma\widetilde{\mathrm{LDE}}_x = \widetilde{\mathrm{LDE}}_x\sigma$.

Observe that the range of $\widetilde{\mathrm{LDE}}_x$ is $\mathbb{F}_{q^d}$ instead of $\mathbb{F}_q$. Thinking of $\mathbb{F}_{q^d}$ as $\mathbb{F}_q^d$, it would be easy to achieve properties $(\tilde{P}1)$ and $(\tilde{P}2)$ by simply repeating LDE_x in each of the d coordinates. The interesting requirement is $(\tilde{P}3)$, and in the appendix we show that is has a clean interpretation in terms of the Fourier Transform of the function $\widetilde{\mathrm{LDE}}_x$.

We can now define the *augmented low-degree extension* and prove that it satisfies the above three properties.

Definition 9 (augmented low-degree extension) *Given $x \in \{0, 1\}^n$, de-*

fine $\widetilde{LDE}_x : \mathbb{F}_{q^d} \to \mathbb{F}_{q^d}$ *as follows:*

$$\widetilde{LDE}_x(y) = \sum_{j=0}^{d-1} LDE_x(\sigma^{-j} y) \cdot \beta^{q^j}. \tag{4}$$

Theorem 10 *Given* $x \in \{0,1\}^n$*, the augmented low-degree extension* $\widetilde{LDE}_x$ *satisfies properties* $(\tilde{P}1)$*,* $(\tilde{P}2)$*, and* $(\tilde{P}3)$*, provided* h *is a prime power,* $h^d > n$*,* q *is a power of* h*, and* $(\log_h q, d) = 1$*. Given* x*,* $\widetilde{LDE}_x$ *can be computed in poly*(q^d) *time.*

PROOF. If follows from the fact that σ is an $\mathbb{F}_q$-linear map that $\mathrm{LDE}_x \sigma^{-j}$ is a d-variate total degree dh polynomial if LDE_x is. Hence $\widetilde{\mathrm{LDE}}_x$ satisfies property $(\tilde{P}1)$.

It is clear that $\pi_0 \widetilde{\mathrm{LDE}}_x = \mathrm{LDE}_x$, and therefore $\widetilde{\mathrm{LDE}}_x$ enjoys property $(\tilde{P}2)$ since LDE_x satisfies the corresponding property $(P2)$.

For property $(\tilde{P}3)$, we observe that for all i,

$$\begin{aligned}\sigma \widetilde{\mathrm{LDE}}_x(\alpha^i) &= \left(\sum_{j=0}^{d-1} \mathrm{LDE}_x(\sigma^{-j} \alpha^i) \beta^{q^j} \right)^q = \sum_{j=0}^{d-1} \mathrm{LDE}_x(\sigma^{-j} \alpha^i) \beta^{q^{j+1}} \\ &= \sum_{j=0}^{d-1} \mathrm{LDE}_x(\sigma^{-j} \sigma \alpha^i) \beta^{q^j} = \widetilde{\mathrm{LDE}}_x(\sigma \alpha^i),\end{aligned}$$

and so $\sigma \widetilde{\mathrm{LDE}}_x = \widetilde{\mathrm{LDE}}_x \sigma$ as required. □

We note that for property $(\tilde{P}3)$, we do not use any specific property of LDE_x – only that it is a function from $\mathbb{F}_{q^d}$ to $\mathbb{F}_q$.

5 The PRG construction

We can now present our PRG construction. As in the SU PRG construction [20] we first build a PRG over a large (non-binary) alphabet, and then convert it to a PRG satisfying the usual definition (Definition 4). In place of equation (1) in that definition, it is sufficient to require that there be no efficient *next-bit predictor* – a function that can predict the i-th bit of output, given the first $i-1$ bits of output, significantly better than random guessing [26]. This alternative requirement turns out to be the correct thing to generalize to define q-ary PRGs:

Definition 11 (q-ary PRG) *Let Σ be an alphabet of q elements. A ρ-q-*ary PRG *for size w is a function $G : \{0,1\}^t \to \Sigma^m$ such that for all $1 \le i \le m$ and all functions $g : \Sigma^{i-1} \to \Sigma^{(\rho^{-2})}$ with size w circuits,*

$$\Pr_y[g(G(y)_{1\ldots i-1}) \ni G(y)_i] < \rho. \tag{5}$$

We are abusing notation slightly – the function f outputs a vector v of ρ^{-2} elements; by "$v \ni G(y)_i$" we mean that $G(y)_i$ occurs as an entry in vector v.

Using an efficiently encodable and list-decodable binary code, we can easily convert q-ary PRGs into ordinary PRGs. In the following definition, $\Delta(\cdot,\cdot)$ is the Hamming distance between two binary strings; i.e., the number of positions in which they differ.

Definition 12 *A binary code $C : \{0,1\}^{\bar{k}} \to \{0,1\}^{\bar{n}}$ is* (ρ,ℓ)-list-decodable *if for all $r \in \{0,1\}^{\bar{n}}$, the set $S_r = \{x : \Delta(C(x), r) \le (1/2-\rho)\bar{n}\}$ has size at most ℓ. The code is* efficiently encodable *if C is computable in time poly($\bar{n}$), and* efficiently list-decodable *if S_r can be computed from r in time poly($\bar{n},\ell$).*

By the Johnson bound (see, e.g., [8]) any family of binary codes with relative distance at least $1/2-\rho^2$ is (ρ,ρ^{-2})-list-decodable. Guruswami and Sudan [7] give several efficiently encodable and efficiently list-decodable (ρ,ρ^{-2})-list-decodable binary codes with blocklength $\bar{n} = \mathrm{poly}(\bar{k},\rho^{-1})$, which is sufficient for our purposes. The following lemma (see, e.g., [23,20]) is by now standard – it essentially applies the Goldreich-Levin hard-core bit reduction to each output element of the q-ary PRG. We provide a short proof for completeness.

Lemma 13 ([23,20]) *Let Σ be an alphabet of q elements, and let $G : \{0,1\}^t \to \Sigma^m$ be a ρ-q-ary PRG for size w. Let $C : \{0,1\}^{\bar{k}=\log q} \to \{0,1\}^{\bar{n}}$ be an efficiently encodable and efficiently list-decodable (ρ,ρ^{-2})-list-decodable code. Then $G' : \{0,1\}^{t+\log\bar{n}} \to \{0,1\}^m$ defined by:*

$$G'(y,j) = C(G(y)_1)_j \circ C(G(y)_2)_j \circ \cdots \circ C(G(y)_m)_j$$

is a $2\rho m$-PRG for size w', provided $w \ge w'\bar{n} + m\bar{n}^{O(1)} + (\bar{n}/\rho)^{O(1)}$.

PROOF. If G' is *not* a $2\rho m$-PRG for size w' then by Yao's argument [26], there exists an i and a *next-bit predictor* $g' : \{0,1\}^{i-1} \to \{0,1\}$ for which

$$\Pr_{y,j}[g'(G'(y,j)_{1\ldots i-1}) = G'(y,j)_i] \ge \frac{1}{2} + 2\rho.$$

Moreover, g' can be implemented by a size $w' + O(1)$ circuit. By an averaging argument, there are at least a ρ fraction of possible y's for which

$$\Pr_j[g'(G'(y,j)_{1\ldots i-1}) = G'(y,j)_i] \geq \frac{1}{2} + \rho. \tag{6}$$

Using g', we now describe a *next-element predictor* $g : \Sigma^{i-1} \to \Sigma^{(\rho^{-2})}$ for G violating property (5). On input $z_1, z_2, \ldots, z_{i-1}$ we compute

$$r_j = g'(C(z_1)_j, C(z_2)_j, \ldots, C(z_{i-1})_j)$$

for all j. Viewing r as an $\bar{n}$-bit string, g outputs S_r (from Definition 12), padding it to have size ρ^{-2} if necessary. Using the fact that C is efficiently encodable and efficient list-decodable, g can be implemented by a size $w'\bar{n} + m\bar{n}^{O(1)} + (\bar{n}/\rho)^{O(1)} \leq w$ circuit.

For each y for which (6) holds, $g(G(y)_{1\ldots i-1})$ computes a set S_r that contains $G(y)_i$; thus g violates property (5), contradicting the fact that G is a q-ary PRG. We conclude that G' must be a $2\rho m$-PRG for size w', as required.

In other words, an $(\epsilon/2m)$-q-ary-PRG for size w with output length m can be converted into a ϵ-PRG for size not much smaller than w with no loss in the output length.

Let $S(x)$ denote the circuit complexity of string $x \in \{0,1\}^n$, viewed as the truth table of a Boolean function on $\log n$ bits. Our main theorem is the following:

Theorem 14 (main) *For every n, d, prime power h, $q = h^c$, string $x \in \{0,1\}^n$ and primitive element α of $\mathbb{F}_{q^d}$ satisfying $h^d > n$ and $(c,d) = 1$, the function $G_x : \mathbb{F}_{q^d} \to \mathbb{F}_{q^d}^m$ defined as follows:*

$$G_x(y) = \widetilde{LDE}_x(\alpha y) \circ \widetilde{LDE}_x(\alpha^2 y) \circ \cdots \circ \widetilde{LDE}_x(\alpha^m y) \tag{7}$$

is a ρ-q^d-ary PRG for size w, provided $S(x) > w \cdot poly(m,q)$ and $q > \Omega(\rho^{-4} h d^4 \log^2 q)$. Furthermore, G_x is computable in $poly(q^d)$ time given access to the truth table of x.

We prove Theorem 14 in the next section. However, given Theorem 14, we can easily prove Theorem 6 by plugging in parameters:

PROOF. [of Theorem 6] Let $x \in \{0,1\}^n$ be the truth table of the hard function f. Notice that if $S(x) \leq \log^{O(1)} n$, then it is sufficient to take G to be the identity function; therefore we assume from now on that $S(x) \geq \log^a n$ for all constants a.

We plug the following parameters into Theorem 14:

- $m = S(x)^\delta$ for a sufficiently small constant $\delta < 1$
- $\rho = \frac{1}{2m^2}$
- d prime between $\frac{\log n}{\log m}$ and $2 \cdot \frac{\log n}{\log m}$
- h prime between m and $2m$
- q smallest power of h larger than $\Omega(m^9 d^4 \log^2 q)$

One can verify that n, d, h and q satisfy the requirements of Theorem 14. By assumption $S(x) \geq \log^{1/\delta} n$, and so $d \leq \log n \leq S(x)^\delta = m$, and it follows that $q = \text{poly}(m)$. Theorem 14 then produces a ρ-q^d-ary PRG for size $w = S(x)/\text{poly}(m, q) = m^{1/\delta - O(1)}$, and by taking δ small enough we have $w \geq m \log^{O(1)} n$ (we are using again that $\log n \leq m$). Since $q = \text{poly}(m)$ we have $q^d = \text{poly}(n)$, and so applying Lemma 13 using a code with blocklength $\bar{n} = \text{poly}(\log(q^d), \rho^{-1})$ gives a $1/m$-PRG for size m as required. The seed length of this PRG is $\log(q^d) + \log \bar{n} = O(\log n)$. $\square$

6 Proof of the main theorem

In this section we give a proof of Theorem 14. Our proof follows the general outline of the list-decoding result of Sudan, Trevisan, and Vadhan [22] as well as the SU PRG proof [20] and indeed resembles the SU PRG proof in many respects. However, there are some critical new details and the notation is quite different from previous accounts. This proof is completely self-contained with the exception of the following lemma of Sudan [21] that allows us to list-decode low-degree univariate polynomials from very noisy data:

Lemma 15 ([21]) *Given* **num** *distinct pairs* (x_i, y_i) *in field* $\mathbb{F}_q$ *and parameters* **agree** *and* **deg** *with*

$$\textbf{agree} > \sqrt{2 \cdot \textbf{deg} \cdot \textbf{num}},$$

there are at most $2 \cdot \textbf{num}/\textbf{agree}$ *degree* **deg** *polynomials* p *such that* $p(x_i) = y_i$ *for at least* **agree** *pairs. Moreover, a list of all such polynomials can be computed in time poly*$(\textbf{num}, \log q)$*.*

We'll also need the following tail-inequality for r-wise independent random variables:

Lemma 16 ([4]) *Let* $r > 4$ *be an even integer. Suppose* $X_1, X_2, \ldots, X_q$ *are*

r-wise independent random variables taking values in $[0,1]$. *Let* $X = \sum X_i$, *and* $A > 0$. *Then:*

$$\Pr[|X - E[X]| \geq A] \leq 8 \cdot \left(\frac{r \cdot E[X] + r^2}{A^2}\right)^{r/2}.$$

The general strategy is as follows. We assume for the purpose of contradiction that we have a *next-element predictor* $g : \mathbb{F}_{q^d}^{i-1} \to \mathbb{F}_{q^d}^{(\rho^{-2})}$, computable by a size w circuit, and violating equation (5). By the symmetry of G_x we may assume that $i = m$, without loss of generality because the predictor can simply ignore some prefix of its input. In the proof we describe an oracle procedure $R : \{0,1\}^a \times [n] \to \{0,1\}$ running in time poly(m, q), and we argue that there exists a short *advice string* $A \in \{0,1\}^a$ for which $R^f(A, i) = x_i$ for all i. Replacing oracle calls with copies of the size w circuit for g, and hardwiring the "good" advice string A, we obtain a circuit of size $w \cdot \text{poly}(m, q)$ computing x, contradicting the hardness of x.

The proof is somewhat lengthy, so we divide it into three parts. In the first part, we show the existence of two *reference curves* that satisfy certain properties. In the second part we show how to perform an *interleaved learning* step, which allows us to learn the evaluations of $\widetilde{\text{LDE}}_x$ at some new locations given the evaluations of $\widetilde{\text{LDE}}_x$ at some old locations. At this point we will have proved the following lemma:

Lemma 17 *There exists an oracle procedure M and degree $\Theta(d^2 \log q)$ curves $C_1 : \mathbb{F}_q \to \mathbb{F}_q^d$ and $C_2 : \mathbb{F}_q \to \mathbb{F}_q^d$, for which the following holds. For all $i \in [q^d - 1]$ and all $\ell \in [d]$, given oracle access to the next-element predictor g, and input:*

$$\left\{\widetilde{LDE}_x(\alpha^{i+1}\sigma^\ell C_k(b)) \ldots \widetilde{LDE}_x(\alpha^{i+m-1}\sigma^\ell C_k(b)) | b \in \mathbb{F}_q, k \in \{1,2\}\right\},$$

M runs in time poly(m, q) and outputs:

$$\left\{\widetilde{LDE}_x(\alpha^{i+m}\sigma^\ell C_k(b)) | b \in \mathbb{F}_q, k \in \{1,2\}\right\}.$$

Informally, M plays the role of an "errorless" next-element predictor, the price for this error-correction being that it must operate "in parallel" at multiple offsets (the points on the reference curves). Shaltiel and Umans [20] prove a very similar lemma that differs in two main respects: they refer to LDE_x, the *ordinary* low-degree extension of x, and there is no mention of σ, the Frobenius automorphism – one might say that their lemma holds only for $\ell = 0$. Indeed, it is precisely the fact that M operates on the augmented low-degree extension $\widetilde{\text{LDE}}_x$ and that it works for all ℓ that enables our procedure R to learn x_i in a small number of prediction steps, utilizing the "jumps" described in the Section 2.2.

The reader familiar with [20] may safely skip Parts 1 and 2, as they address only technical modifications of the proofs in that paper required to prove Lemma 17. The lemma is then used essentially as a black box in the third part of the proof, in which we describe how to use a short sequence of interleaved learning steps to learn x_i, given i. This is the main formal description of the use of the augmented low-degree extension.

6.1 Part 1: reference curves

Set $r = \Theta(d^2 \log q)$. We show by the probabilistic method that there exist "good" degree r *reference curves* $C_1 : \mathbb{F}_q \to \mathbb{F}_q^d$ and $C_2 : \mathbb{F}_q \to \mathbb{F}_q^d$. By degree r we mean that $\pi_j C_1$ and $\pi_j C_2$ are degree r univariate polynomials, for all $j \in [d]$. Using Property $(\tilde{P}1)$ and the fact that α and σ are $\mathbb{F}_q$-linear maps from $\mathbb{F}_q^d$ to itself, we see that

$$p(b) = \pi_j(G_x(\alpha^i \sigma^\ell C_1(b))_m) = \pi_j \widetilde{\mathrm{LDE}}_x(\alpha^{m+i} \sigma^\ell C_1(b))$$

is a degree $\leq rdh$ univariate polynomial for all $i \in [q^d - 1]$, all $j \in [d]$ and all $\ell \in [d]$; the same holds after replacing C_1 with C_2.

For C_1 and C_2 to be "good," we require that they satisfy the following three properties, for all $i \in [q^d - 1]$, all $j \in [d]$, and all $\ell \in [d]$:

(1) the success rate of the next-element predictor g along any "shift" of C_1 or C_2 is close to its overall success rate ρ:

$$\Pr_{b \in \mathbb{F}_q} [g(G_x(\alpha^i \sigma^\ell C_1(b))_{1\ldots m-1}) \ni G_x(\alpha^i \sigma^\ell C_1(b))_m] \geq \rho/2,$$

and the same holds after replacing C_1 with C_2.

(2) If $T^{(1)}_{i,j,\ell}$ is a set of at most $4\rho^{-3}$ degree $\leq rdh$ univariate polynomials determined by C_1, and including

$$p(b) = \pi_j(G_x(\alpha^i \sigma^\ell C_1(b))_m)$$

then for all $p' \in T^{(1)}_{i,j,\ell}$ such that $p' \not\equiv p$, there exists $b \in \left\{b | \sigma^\ell C_1(b) = \alpha^{-1} \sigma^\ell C_2(b)\right\}$ for which $p'(b) \neq p(b)$.

(3) If $T^{(2)}_{i,j,\ell}$ is a set of at most $4\rho^{-3}$ degree $\leq rdh$ univariate polynomials determined by C_2, and including

$$p(b) = \pi_j(G_x(\alpha^i \sigma^\ell C_2(b))_m)$$

then for all $p' \in T^{(2)}_{i,j,\ell}$ such that $p' \not\equiv p$, there exists $b \in \{b | C_1(b) = C_2(b)\}$ for which $p'(b) \neq p(b)$.

To show that "good" curves C_1 and C_2 exist, we pick them randomly as in the SU PRG [20]; the argument that they are "good" with high probability then follows [20] closely. We describe their argument below.

Set $r' = r/(d+1)$ and pick r random elements of $\mathbb{F}_q$ (by the lower bound on q, these are distinct with high probability):

$$b_{0,1}, \ldots, b_{0,r'}, b_{1,1}, \ldots, b_{1,r'}, \ldots, b_{d,1}, \ldots, b_{d,r'}$$

and r random vectors in $\mathbb{F}_q^d$:

$$v_{0,1}, \ldots, v_{0,r'}, v_{1,1}, \ldots, v_{1,r'}, \ldots, v_{d,1}, \ldots, v_{d,r'}.$$

Curves C_1 and C_2 are defined to be the degree $(r-1)$ curves for which $C_1(b_{i,j}) = v_{i,j}$ for all i, j and $C_2(b_{i,j}) = v_{i,j}$ for $i = 0$ and all j, and $C_2(b_{i,j}) = \alpha^{q^i} v_{i,j}$ for all $i > 0$ and all j. In the next three claims, we show that properties (1), (2), and (3) all hold with very high probability.

Claim 18 *For fixed i, j and ℓ, with probability at least $1 - \frac{1}{6d^2q^d}$ over our choice of C_1 and C_2, property (1) above holds.*

PROOF. The set $\left\{\alpha^i \sigma^\ell C_1(b) | b \in \mathbb{F}_q\right\}$ is an r-wise independent set of points in $\mathbb{F}_q^d$, and the same holds after replacing C_1 with C_2. This follows from the fact that α and σ are both bijective mappings from $\mathbb{F}_{q^d}$ to itself.

Define X_b to be the indicator random variable for the event

$$g(G_x(\alpha^i \sigma^\ell C_1(b))_{1\ldots m-1}) \ni G_x(\alpha^i \sigma^\ell C_1(b))_m;$$

i.e., the event that the next-element predictor g succeeds in predicting position $\alpha^i \sigma^\ell C_1(b)$. Define $X = \sum_{b \in \mathbb{F}_q} X_b$, and notice that $E[X] > \rho q$.

Applying Lemma 16, we get that:

$$\Pr[X < \rho q/2] \leq 8 \cdot \left(\frac{4(r\rho q + r^2)}{(\rho q)^2}\right)^{r/2} \leq \left(\frac{O(r)}{\rho q}\right)^{r/2} \leq \frac{1}{12d^2q^d},$$

where the probability is over the choice of C_1, and we are using the fact that $\rho q \geq q^{3/4} \geq r \geq 2$. The same bound holds after replacing C_1 with C_2, and so property (1) holds with probability at least $1 - 2(\frac{1}{12d^2q^d})$ as claimed. $\square$

Claim 19 *For fixed i, j and ℓ, with probability at least $1-\frac{1}{6d^2q^d}$ over our choice of C_1 and C_2, property (2) above holds.*

PROOF. For this proof it is convenient to imagine picking C_1 and C_2 by *first* picking C_1 to be a random degree $r-1$ curve, and *then* picking the random b's on this curve which in turn determine C_2. This allows us to regard the set $T^{(1)}_{i,j,\ell}$, which depends on C_1, as fixed when we pick the b's.

The set $S = \{b_{d-\ell,k} | k \in [r']\}$ is a set of r' random values in $\mathbb{F}_q$, and for every $b \in S$, we have $\sigma^\ell C_1(b) = \alpha^{-1}\sigma^\ell C_2(b)$. To see this, observe that:

$$\alpha^{-1}\sigma^\ell C_2(b_{d-\ell,k}) = \alpha^{-1}\sigma^\ell \alpha^{q^{d-\ell}} v_{d-\ell,k} = \alpha^{-1}\alpha^{q^d}\sigma^\ell v_{d-\ell,k} = \sigma^\ell v_{d-\ell,k}$$

which is exactly $\sigma^\ell C_1(b_{d-\ell,k})$ for all $k \in [r']$.

Now consider a fixed $p' \in T^{(1)}_{i,j,\ell}$ such that $p' \not\equiv p$. Since two degree $\leq rdh$ polynomials can agree on at most rdh points, we have:

$$\Pr[\forall b \in S \;\; p'(b) = p(b)] \leq \left(\frac{rdh}{q}\right)^{r'} < \frac{1}{(4\rho^{-3}) \cdot 6d^2q^d},$$

using the lower bound on q in the statement of the theorem. By taking a union bound, we have that the probability that there *exists* a "bad" $p' \in T^{(1)}_{i,j,\ell}$ is at most $\frac{1}{6d^2q^d}$, and the claim follows. □

Claim 20 *For fixed i, j and ℓ, with probability at least $1-\frac{1}{6d^2q^d}$ over our choice of C_1 and C_2, property (3) above holds.*

PROOF. As in the proof of Claim 19 we imagine picking C_1 and C_2 by *first* picking C_2 to be a random degree $r-1$ curve, and *then* picking the random b's. This allows us to regard $T^{(2)}_{i,j,\ell}$, which depends on C_2, as fixed when we pick the b's.

The set $S = \{b_{0j} | j \in [r']\}$ is a set of r' random values in $\mathbb{F}_q$, and for every $b \in S$, we have $C_1(b) = C_2(b)$. The remainder of the proof is identical to the proof of Claim 19. □

Taking a union bound, the probability that there *exist* i, j, ℓ for which one of the three properties fails to hold is at most $3d^2q^d \cdot \frac{1}{6d^2q^d} = 1/2$. Thus, there exist "good" curves C_1 and C_2 for which properties (1), (2), and (3) hold simultaneously for all i, j and ℓ; from now on, we fix a pair of "good" C_1 and C_2 as our *reference curves.*

For all $i \in [q^d - 1]$ and all $\ell \in [d]$, we describe how to learn the values of

$$G_x(\alpha^i \sigma^\ell C_1(b))_m \text{ and } G_x(\alpha^i \sigma^\ell C_2(b))_m \ \ \forall b \in \mathbb{F}_q$$

given the values of

$$G_x(\alpha^i \sigma^\ell C_1(b))_{1\ldots m-1} \text{ and } G_x(\alpha^i \sigma^\ell C_2(b))_{1\ldots m-1} \ \ \forall b \in \mathbb{F}_q$$

in $\mathrm{poly}(m, q)$ time, with oracle access to the next-element predictor g. The algorithm follows:

- compute a set of "predicted" values for all $j \in [d]$ and $b \in \mathbb{F}_q$, using dq calls to g:

$$S^{(1)}_{j,b} = \left\{ \pi_j(g(G_x(\alpha^i \sigma^\ell C_1(b))_{1\ldots m-1})_s) | s \in [\rho^{-2}] \right\}.$$

- For each $j \in [d]$, notice that $p^{(1)}_j(b) = \pi_j(G_x(\alpha^i \sigma^\ell C_1(b))_m)$ is a univariate polynomial of degree $\leq rdh$, and by property (1) in Part 1, it agrees with at least $(\rho/2)q$ of the (point, evaluation) pairs $\left\{(b, c) : b \in \mathbb{F}_q, c \in S^{(1)}_{j,b}\right\}$. Using Lemma 15 (with $\mathbf{num} = q\rho^{-2}, \mathbf{agree} = (\rho/2)q$, and $\mathbf{deg} = rdh$) we compute for each j the set $T^{(1)}_{i,j,\ell}$ of at most $4\rho^{-3}$ degree $\leq rdh$ polynomials with this agreement.
- We know the values of

$$G_x(\alpha^i \sigma^\ell C_2(b))_{m-1} = G_x(\alpha^{i-1} \sigma^\ell C_2(b))_m$$

  for all $b \in \mathbb{F}_q$. This means that we know $p^{(1)}_j(b) = \pi_j \widetilde{\mathrm{LDE}}(\alpha^{i+m} \sigma^\ell C_1(b))$ for all

$$b \in \left\{ b | \alpha^{i+m} \sigma^\ell C_1(b) = \alpha^{i+m-1} \sigma^\ell C_2(b) \right\} = \left\{ b | \sigma^\ell C_1(b) = \alpha^{-1} \sigma^\ell C_2(b) \right\}.$$

  We now look for a unique polynomial in $T^{(1)}_{i,j,\ell}$ that agrees with the evaluations of $p^{(1)}_j(b)$ at this special set of b's. Property (2) in Part 1 ensures that $p^{(1)}_j(b)$ is the *unique* polynomial in $T^{(1)}_{i,j,\ell}$ with this agreement. We do this for each j, at which point we have $p^{(1)}_j(b)$ for all j, and then we can assemble them to obtain $G_x(\alpha^i \sigma^\ell C_1(b))_m = \sum_{j=0}^{d-1} p^{(1)}_j(b) \beta^{q^j}$. We have therefore successfully learned $G_x(\alpha^i \sigma^\ell C_1(b))_m$.
- Repeat with respect to C_2; that is, compute sets of predicted values

$$S^{(2)}_{j,b} = \left\{ \pi_j(g(G_x(\alpha^i \sigma^\ell C_2(b))_{1\ldots m-1})_s) | s \in [\rho^{-2}] \right\}$$

and use Lemma 15 to compute sets of candidate polynomials $T^{(2)}_{i,j,\ell}$ consisting of all degree $\le rdh$ polynomials that agree on at least $(\rho/2)q$ of the (point, evaluation) pairs $\left\{(b,c) : b \in \mathbb{F}_q, c \in S^{(2)}_{j,b}\right\}$. By property (1) in Part 1, $p^{(2)}_j(b) = \pi_j(G_x(\alpha^i\sigma^\ell C_2(b))_m)$ is contained in $T^{(2)}_{i,j,\ell}$.

We know the values of $G_x(\alpha^i\sigma^\ell C_1(b))_m$ for all $b \in \mathbb{F}_q$. This means that we know $p^{(2)}_j(b) = \pi_j\widetilde{\mathrm{LDE}}(\alpha^{i+m}\sigma^\ell C_2(b))$ for all

$$b \in \left\{b|\alpha^{i+m}\sigma^\ell C_1(b) = \alpha^{i+m}\sigma^\ell C_2(b)\right\} = \{b|C_1(b) = C_2(b)\}.$$

We look for a unique polynomial in $T^{(2)}_{i,j,\ell}$ that agrees with the evaluations of $p^{(2)}_j(b)$ on this set of b's. Property (3) in Part 1 ensures that $p^{(2)}_j(b)$ is the *unique* polynomial in $T^{(2)}_{i,j,\ell}$ with this agreement. We do this for each j, at which point we can obtain $G_x(\alpha^i\sigma^\ell C_2(b))_m = \sum_{j=0}^{d-1} p^{(2)}_j(b)\beta^{q^j}$. We have therefore successfully learned $G_x(\alpha^i\sigma^\ell C_2(b))_m$.

Altogether, the above algorithm requires poly(m,q) time with oracle access to g; we have therefore proved Lemma 17.

6.3 Part 3: a sequence of interleaved learning steps

Given $i \in [n]$, we show how to use a short sequence of at most dmq interleaved learning steps to learn x_i, given the values of

$$G_x(C_1(b))_{1\ldots m-1} \text{ and } G_x(C_2(b))_{1\ldots m-1} \quad \forall b \in \mathbb{F}_q.$$

We will use $V_{i,\ell}$ as shorthand for:

"the values of $G_x(\alpha^{i-m}\sigma^\ell C_1(b))_m$ and $G_x(\alpha^{i-m}\sigma^\ell C_2(b))_m$ for all $b \in \mathbb{F}_q$."

Using the definition of G_x, another way to describe $V_{i,\ell}$ is "the evaluations of $\widetilde{\mathrm{LDE}}_x(\alpha^i\sigma^\ell C_1(b))$ and $\widetilde{\mathrm{LDE}}_x(\alpha^i\sigma^\ell C_2(b))$ for all $b \in \mathbb{F}_q$."

Since $\alpha^{q^d-1} = \alpha^0$ and $\sigma^d = \sigma^0$, we consider the subscripts of $V_{i,\ell}$ modulo $q^d - 1$ and d, respectively. Notice that for all i and all ℓ a single interleaved learning step allows us to learn $V_{i+m,\ell}$ from $V_{i+1,\ell}, V_{i+2,\ell}, \ldots, V_{i+m-1,\ell}$. This statement is exactly Lemma 17.

The key consequence of Property $(\tilde{P}3)$ that we use is that $V_{iq,\ell+1}$ can be efficiently computed from $V_{i,\ell}$, and vice versa, for all i and ℓ. This follows from:

$$\begin{aligned}\sigma(G_x(\alpha^{i-m}\sigma^\ell C_1(b))_m) &= \sigma\widetilde{\mathrm{LDE}}_x(\alpha^i\sigma^\ell C_1(b)) = \widetilde{\mathrm{LDE}}_x(\sigma(\alpha^i\sigma^\ell C_1(b))) \\ &= \widetilde{\mathrm{LDE}}_x(\alpha^{iq}\sigma^{\ell+1}C_1(b)) = G_x(\alpha^{iq-m}\sigma^{\ell+1}C_1(b))_m\end{aligned}$$

for all $b \in \mathbb{F}_q$, which also holds after replacing C_1 with C_2. That is, to compute each evaluation in $V_{iq,\ell+1}$ we take the corresponding evaluation in $V_{i,\ell}$ and exponentiate by q (i.e., apply σ); to compute each evaluation in $V_{i,\ell}$ we apply σ^{-1} to the corresponding evaluation in $V_{iq,\ell+1}$.

We now proceed to describe the algorithm. Let $c \in [q^d - 1]$ be such that $\alpha^c = C_1(0)$, and set $t = mq$. Set $z = \ell(i) - c - 1$, and write it in its q-ary representation: $z = \sum_{\ell=0}^{d-1} z_\ell q^\ell$.

- Set $a_0 = 0$. Notice that $V_{a_0+1,0}, V_{a_0+2,0}, \ldots, V_{a_0+(m-1),0}$ are given. Use $t-m+1$ successive interleaved learning steps to learn:

  $$V_{a_0+m,0}, V_{a_0+(m+1),0}, \ldots, V_{a_0+t,0}.$$

- For $\ell = 1, 2, \ldots, d-1$:
  - Observe that in the previous step we have learned $V_{a_{\ell-1}+1,-(\ell-1)}, \ldots V_{a_{\ell-1}+t,-(\ell-1)}$
  - Set $a_\ell = (a_{\ell-1} + z_{\ell-1})q^{d-1}$. Notice that

    $$V_{a_\ell+1,-\ell}, V_{a_\ell+2,-\ell}, \ldots, V_{a_\ell+(m-1),-\ell}$$

    can be easily computed from

    $$V_{a_{\ell-1}+z_{\ell-1}+q,-(\ell-1)}, V_{a_{\ell-1}+z_{\ell-1}+2q,-(\ell-1)}, \ldots, V_{a_{\ell-1}+z_{\ell-1}+(m-1)q,-(\ell-1)},$$

    which we know from the previous step (using the fact that $0 \le z_{\ell-1} < q$). Use $t - m + 1$ successive interleaved learning steps to learn:

    $$V_{a_\ell+m,-\ell}, V_{a_\ell+(m+1),-\ell}, \ldots, V_{a_\ell+t,-\ell}.$$

- Set $a_d = (a_{d-1} + z_{d-1})q^{d-1}$. Notice that $V_{a_d+1,-d}$ can be efficiently computed from $V_{a_{d-1}+z_{d-1}+q,-(d-1)}$ which we know from the previous step. Output $\pi_0(G_x(\alpha^{a_d+1-m}C_1(0))_m)$, whose value is contained in $V_{a_d+1,-d}$.

By an easy induction on ℓ, we see that for $\ell = 1, 2, \ldots d$,

$$a_\ell = \sum_{j=d-\ell}^{d-1} z_{j-d+\ell}q^j,$$

where all integers are considered modulo $q^d - 1$. We conclude that we have output x_i, since:

$$\begin{aligned}\pi_0(G_x(\alpha^{a_d+1-m}C_1(0))_m) &= \pi_0\widetilde{\mathrm{LDE}}_x(\alpha^{a_d+1}C_1(0)) = \pi_0\widetilde{\mathrm{LDE}}_x(\alpha^{z+1}C_1(0)) \\ &= \pi_0\widetilde{\mathrm{LDE}}_x(\alpha^{\ell(i)}) = x_i,\end{aligned}$$

using Property ($\tilde{P}2$) for the last equality. The total number of interleaved learning steps we used was $d(t-m+1) \leq dmq$. Each interleaved learning step requires poly(m, q) time with oracle access to g, so altogether we have given a procedure (that requires some non-uniform advice) to compute x_i given i, that runs in time poly(m, q) with oracle access to the next-element predictor g.

6.4 Putting it all together

It remains to describe the advice required for the above procedure to get started. We need to provide:

- degree r curves C_1 and C_2 satisfying the three properties listed in subsection 6.1. Each can be described by r coefficients in $\mathbb{F}_q^d$.
- values of

  $$G_x(C_1(b))_{1\ldots m-1} \text{ and } G_x(C_2(b))_{1\ldots m-1}$$

  for all $b \in \mathbb{F}_q$. As we have observed, these can be described as $2d(m-1)$ different degree $\leq rdh$ univariate polynomials, each requiring rdh coefficients in $\mathbb{F}_q$.
- integer $c \in [q^d - 1]$ for which $\alpha^c = C_1(0)$, and a few miscellaneous items for field arithmetic: the basis for $\mathbb{F}_{q^d}$ over $\mathbb{F}_q$, the irreducible polynomial used to construct the extension field, and the primitive element α.

Altogether the advice string has length $(2rd + 2rd^2h(m-1) + O(d)) \log q =$ poly(m, q). The overall running time with oracle access to g is also poly(m, q), so after substituting our size w circuit for g, and hardwiring the advice, we get a circuit of size at most $w \cdot$ poly(m, q) that computes x_i given i, contradicting the hardness of x. This concludes the proof of the main theorem.

7 Conclusions

We have given an optimal (up to a polynomial) construction of PRGs from hard functions. The basic framework of Ta-Shma, Zuckerman and Safra [23], as extended by Shaltiel and Umans [20], has now been used to construct quite a variety of derandomization objects – extractors for a wide range of parameters, optimal HSGs, optimal non-deterministic HSGs (from single-valued nondeterministic hardness, as in Milterson and Vinodchandran's work [18]), and in the present paper, optimal PRGs.

We remark that the extension to non-deterministic HSGs and PRGs [20] also applies to our PRG construction, so a further consequence of this paper is a construction of optimal non-deterministic PRGs. The main idea of Shaltiel-Umans regarding nondeterministic PRGs permits constructions which utilize next-element predictors along random curves (as we do) to operate in the nondeterministic setting. A nondeterministic PRG fools small *nondeterministic* circuits, using a function that is hard for *single-valued nondeterministic* circuits. We refer to the journal version of Shaltiel-Umans [20] for the relevant definitions. For completeness we state the new theorem we obtain (compare to Theorem 6):

Theorem 21 (non-deterministic PRG) *Given a function* $f : \{0,1\}^{\log n} \to \{0,1\}$ *with* single-valued nondeterministic *circuit complexity at least* s*, one can construct an* $1/m$-nondeterministic *PRG* $G : \{0,1\}^{O(\log n)} \to \{0,1\}^m$ *for size* m *with* $m = s^{\Omega(1)}$*; moreover,* G *can be computed in* $n^{O(1)}$ *time given access to the truth table of* f*.*

In conclusion, it seems that even quite elementary algebra is a useful tool in this arena, and as noted in the introduction it would be interesting to see if the current set of ideas with some new tricks can produce optimal extractors, which remain an important and so-far elusive goal in this area.

8 Acknowledgements

We thank Henry Cohn and Ronen Shaltiel for useful discussions. We also thank the anonymous referees for numerous helpful comments.

References

[1] A. Andreev, A. Clementi, and J. Rolim. A new general derandomization method. *J. ACM*, 45(1), 1998.

[2] A. E. Andreev, A. E. F. Clementi, J. D. P. Rolim, and L. Trevisan. Weak random sources, hitting sets, and BPP simulations. In *38th Annual Symposium on Foundations of Computer Science*, pages 264–272, Miami Beach, Florida, 20–22 Oct. 1997. IEEE.

[3] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3(4):307–318, 1993.

[4] M. Bellare and J. Rompel. Randomness-efficient oblivious sampling. In *35th Annual Symposium on Foundations of Computer Science*, 1994.

[5] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.*, 13(4):850–864, Nov. 1984.

[6] O. Goldreich, S. Vadhan, and A. Wigderson. Simplified derandomization of BPP using a hitting set generator. Technical Report TR00-004, Electronic Colloquium on Computational Complexity, January 2000.

[7] V. Guruswami and M. Sudan. List decoding algorithms for certain concatenated codes. In *Proceedings of the 32nd ACM Symposium on Theory of Computing*, 2000.

[8] V. Guruswami and M. Sudan. Extensions to the Johnson bound. Manuscript, February 2001.

[9] R. Impagliazzo. Hard-core distributions for somewhat hard problems. In *36th Annual Symposium on Foundations of Computer Science*, pages 538–545, Milwaukee, Wisconsin, 23–25 Oct. 1995. IEEE.

[10] R. Impagliazzo, V. Kabanets, and A. Wigderson. In search of an easy witness: Exponential time vs. probabilistic polynomial time. In *Sixteenth Annual IEEE Conference on Computational Complexity*, pages 1–12, 2001. To appear in *Journal of Computer and System Sciences*.

[11] R. Impagliazzo, R. Shaltiel, and A. Wigderson. Near-optimal conversion of hardness into pseudo-randomness. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, 1999.

[12] R. Impagliazzo, R. Shaltiel, and A. Wigderson. Extractors and pseudo-randomn generators with optimal seed-length. In *Proceedings of the Thirty-second Annual ACM Symposium on the Theory of Computing*, 21–23 May 2000.

[13] R. Impagliazzo and A. Wigderson. $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 220–229, El Paso, Texas, 4–6 May 1997.

[14] V. Kabanets. Easiness assumptions and hardness tests: Trading time for zero error. *Journal of Computer and System Sciences*, 63(2):236–252, 2001.

[15] A. R. Klivans and D. van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*, 1999. To appear in *SIAM Journal on Computing*.

[16] S. Lang. *Algebra (third edition)*. Addison-Wesley, 1993.

[17] C. J. Lu. Derandomizing Arthur-Merlin games under uniform assumptions. *Computational Complexity*, 10(3):247–259, 2001.

[18] P. B. Miltersen and N. V. Vinodchandran. Derandomizing Arthur-Merlin games using hitting sets. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, 1999.

[19] N. Nisan and A. Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, Oct. 1994.

[20] R. Shaltiel and C. Umans. Simple extractors for all min-entropies and a new pseudo-random generator. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science*, pages 648–657, 2001.

[21] M. Sudan. Decoding of Reed Solomon codes beyond the error-correction bound. *Journal of Complexity*, 13, 1997.

[22] M. Sudan, L. Trevisan, and S. Vadhan. Pseudorandom generators without the XOR lemma. *Journal of Computer and System Sciences*, 62(2):236–266, Mar. 2001.

[23] A. Ta-Shma, D. Zuckerman, and S. Safra. Extractors from Reed-Muller codes. In *Proceedings of the 42th Annual IEEE Symposium on Foundations of Computer Science*, 2001.
[24] L. Trevisan. Extractors and pseudorandom generators. *Journal of the ACM*, 48(4):860–879, 2001.
[25] R. Urbanke. Modern coding theory – SS2001. Technical Report DSC-LTHC, EPFL, 2001. Available from
`http://lthcwww.epfl.ch/content.php?title=coding2001`
[26] A. C. Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science*, pages 80–91, Chicago, Illinois, 3–5 Nov. 1982. IEEE.

A Fourier interpretation of property ($\tilde{\mathbf{P}}\mathbf{3}$)

In this appendix we observe that property ($\tilde{P}3$), which is exactly what we need for our PRG construction, has a clean mathematical interpretation; we emphasize, however, that this characterization is not required anywhere in the body of the paper.

We essentially reproduce the following definition and theorem from a presentation of BCH codes [25], which can be seen as the *Fourier transforms* of functions satisfying property ($\tilde{P}3$) and one additional property. In contrast, our augmented low-degree extension code satisfies property ($\tilde{P}3$) and two additional properties; to our knowledge this is the extent of its connection to BCH codes.

Definition 22 (finite field Fourier Transform) *Let $\mathbb{F}$ be a field of n elements and α a primitive element of $\mathbb{F}$. Given a function $f : \mathbb{F}^* \to \mathbb{F}^*$, the Fourier transform of f is given by:*

$$\hat{f}(\alpha^j) = \sum_{i=1}^{n-1} f(\alpha^i)\alpha^{ij} \tag{A.1}$$

and the Inverse Fourier Transform is given by:

$$f(\alpha^i) = \frac{1}{n-1}\sum_{j=1}^{n-1} \hat{f}(\alpha^j)\alpha^{-ij}. \tag{A.2}$$

Theorem 23 *A function $f : \mathbb{F}^*_{q^d} \to \mathbb{F}^*_{q^d}$ satisfies $\sigma f = f\sigma$ if and only if the image of $\hat{f}$ is contained in $\mathbb{F}_q$.*

PROOF. We rely repeatedly on the fact that the elements of the base field $\mathbb{F}_q$ are exactly those elements $x \in \mathbb{F}_{q^d}$ for which $x^q = x$.

($\Rightarrow$) Suppose that $\sigma f = f\sigma$. Then we have for all j,

$$\begin{aligned}(\hat{f}(\alpha^j))^q &= \left(\sum_{i=1}^{q^d-1} f(\alpha^i)\alpha^{ij}\right)^q = \sum_{i=1}^{q^d-1} \sigma f(\alpha^i)\alpha^{ijq} \\ &= \sum_{i=1}^{q^d-1} f(\sigma\alpha^i)\alpha^{ijq} = \sum_{i=1}^{q^d-1} f(\alpha^{iq})\alpha^{(iq)j} = \hat{f}(\alpha^j),\end{aligned}$$

which implies that $\hat{f}(\alpha^j) \in \mathbb{F}_q$ — the last equality holds because

$$\left\{iq | i = 1, 2, \ldots, q^d - 1\right\} \equiv \left\{1, 2, \ldots, q^d - 1\right\} \pmod{q^d - 1},$$

i.e., α^q generates $\mathbb{F}^*_{q^d}$ if α does.

($\Leftarrow$) Suppose that $\hat{f}(\alpha^j) \in \mathbb{F}_q$ for all j. Then we have for all i:

$$\begin{aligned}\sigma f(\alpha^i) &= \left(\frac{1}{q^d-1}\sum_{j=1}^{q^d-1} \hat{f}(\alpha^j)\alpha^{-ij}\right)^q = \left(\frac{1}{q^d-1}\right)^q \sum_{j=1}^{q^d-1} \hat{f}(\alpha^j)^q\alpha^{-(iq)j} \\ &= \frac{1}{q^d-1}\sum_{j=1}^{q^d-1} \hat{f}(\alpha^j)\alpha^{-(iq)j} = f(\alpha^{iq}) = f(\sigma\alpha^i),\end{aligned}$$

and hence $\sigma f = f\sigma$. $\square$