Trinomial Hardness and PRG's

Casmali Lopez, Paisios Woodcock

June 2022

1 Introduction

Suppose one is asked to pick a random number. For simplicity, say we are working in binary and that we know how many digits this random number should have. Thus generating a random number is equivalent to generating a sufficiently random sequence of bits – zeros and ones. To choose such a 'random' sequence requires both a standard of randomness and also a deterministic method of producing a sequence to fit that standard. Pseudo-random generators (PRG's) are the method, (so named because they are deterministic and hence not truly random) and the standard they meet is defined by unpredictability. Intuitively, if it is not computationally feasible to predict what the next bit of a sequence is (given the first however many), then we can say it is unpredictable and thus, in that sense, pseudo-random.

The topic of this paper is to generalize the results of a paper by Blum and Micali [BM84], in which rigorous parameters are given for these ideas of being hard to predict, pseudo-random computationally infeasible and so on. Their work focuses on the assumed hardness of the Discrete Logarithm Problem, but we provide here a more abstract approach for how their style of PRG's, one which attempts to turn a given function (meeting some criteria) into such a random generator. We also provide more general ideas for assessing how unpredictable a PRG is, or how compututationally infeasible it is to predict it, as tools for assessing the value of a given PRG. We also assess the feasibility of using a trinomial as a PRG generator utilizing some conditional results as well as some experimental results. Finally, we note side results that were uncovered during research, future directions and related open questions.

Note that, for shorthand, when $F = \{f_i\}$ is a set of functions, we say a function g is on $O(F)$ if $\exists f_i \in F$ such that $g \in O(f_i)$: and similarly for o, ω, and Ω notation.

2 Background

2.1 Generalized Pseudo-Random-Generators

Definition 2.1. A $\Phi\Gamma\Upsilon$ pseudo-random bit generator ($\Phi\Gamma\Upsilon-$PRG), defined for hardness class Γ and families of functions Φ, Υ, is a family of functions $G := \{G_n : \{0,1\}^n \to \{0,1\}^{Q(n)}\}$, with $Q : \mathbb{N} \to \mathbb{N}$ an increasing function, with the property that setting $(y_1, ..., y_{Q(n)}) := G_n(x_1, ..., x_n)$ satisfies this: for every $i \in \{0, ..., Q(n) - 1\}$, every algorithm $A \in \Gamma$, and every $\phi \in \Phi$,

$$|Prob_{(x_1,...,x_n)\in\{0,1\}^n}[A(y_1, ..., y_i) = y_{i+1}] - \frac{1}{2}| \in o(\frac{1}{\phi(n)}),$$

where each y_i is computable in time on $O(\Upsilon)$.

Here we assume the input is an n-bit binary integer x, rather than a sequence. We associate with any Γ a set of functions F, where Γ is the set of all algorithms computable in time in the order of some function in F (on n). Now, because circuits are algorithms of arbitrary intricacy, or can be efficiently transformed into such, the following is an equivalent way of defining a PRG.

Definition 2.2. A $F\Phi\Upsilon$ pseudo-random bit generator ($F\Phi\Upsilon-$PRG), where F, Φ, and Υ are sets of functions, is a family of functions $G := \{G_n : \{0,1\}^n \to \{0,1\}^{Q(n)}\}$, with $Q : \mathbb{N} \to \mathbb{N}$ an increasing function, with the property that setting $(y_1, ..., y_{Q(n)}) := G_n(x_1, ..., x_n)$ satisfies this: for every $i \in \{0, ..., Q(n) - 1\}$, every family o circuits $C = \{C_n\}$ with number of circuits on $O(F)$, and every $\phi \in \Phi$,

$$|Prob_{(x_1,...,x_n)\in\{0,1\}^n} C_n[y_1, ..., y_i] = y_{i+1}] - \frac{1}{2}| \in o(\frac{1}{\phi(n)}),$$

where each y_i is computable in time on $O(\Upsilon)$.

2.2 α, β-Periodic

The outputs of a $\Gamma\Phi\Upsilon-$PRG we will call $\Gamma\Phi\Upsilon$-PRG sequences. All such sequences are ultimately periodic [cite(BM)]. Let α and β be integers. A $\Gamma\Phi\Upsilon$-PRG sequence is (α, β)-periodic if it becomes periodic, with period length less than β, after at most α bits. To be unpredictable, we need $\alpha + \beta < Q(n)$, the length of the outputted message, lest an adversary be able to often predict the next bit by using the periodicity of the sequence. For simplicity, we will assume all PRG sequences have this property of $\alpha + \beta < Q(n)$.

2.3 Predicate

Let S_n be a subset of the n-bit integers, with $i \in S_n$. Let D_i be a subset of the integers with at most n bits. We call B a set of predicates if $B = \{B_i : D_i \to \{0,1\} | i \in S_n, n \in N\}$. Essentially, each B_i assigns elements of D_i a

binary value. Every element of D_i is represented with n bits, if necessary by prefixing 0's. We call a given i an n-bit input "seed".

2.4 Inputs

Let $I = \{I_n | n \in N\}$ denote the set of all inputs to a set of predicates B, where $I_n = \{(i, x) | i \in S_n \text{ and } x \in D_i\}$. An element in I_n is called an input of size n.

2.5 υ-Accessible

We say a predicate B is υ-accessible if there exists constant c_0 and probabilistic algorithm A such that, on input length n, A has expected run in time $\upsilon(n)$, A outputs "?" with probability $(\frac{1}{2})^{c_2}$, and whenever A does not output "?", it outputs a pair $(i, x) \in I_n$ with uniform probability among elements of I_n.

2.6 $\Phi_1\Phi_2$-Unapproximable

Let B be a set of predicates and Φ_1, Φ_2 be sets of functions. Let $C_n^{\phi_1}$ denote the size of the smallest circuit $C = C[\cdot, \cdot]$ that computes $B_i(x)$ correctly for at least a fraction of $\frac{1}{2} + \frac{1}{\phi_1(n)}$ of all $(i, x) \in I_n$. Such a circuit C is said to $\frac{1}{\phi_1(n)}$-approximate B (where $\phi_1 \in \Phi_1$). We say B is $\Phi_1\Phi_2$-unapproximable if, for any $\phi_1 \in \Phi_1$, $C_n^{\phi_1}$ is not on the order of any function in Φ_2.

Unapproximability is thus defined in terms of resistance to being efficiently solved (with a certain degree of accuracy) by circuits, but this easily generalizes to algorithms. Given a hardness class Γ of algorithms with run time on the order of some function in Φ_1, we say this: B is $\Phi\Gamma_2$-unapproximable if no algorithm in Γ can solve

Put another way, such a B is $\Phi_1\Phi_2$-unapproximable if, for all $\phi_1 \in \Phi_1$ and $\phi_2 \in \Phi_2$, there exists an infinite set of integers T such that for all $n \in T$, any circuit calculating $B_i(x)$ correctly for at least a fraction $\frac{1}{2} + \frac{1}{\phi_1(n)}$ of all $(i, x) \in I_n$ has more than $\phi_2(n)$ gates.

3 Sufficient Conditions for a $\Phi\Gamma\Upsilon$-PRG

Using the above, we generalize Blum and Micali's Theorem 2 [BM84] to provide measurements on how good a given PRG is and what aspects of a predicate and function are sufficient for the formulation of a PRG.

[Theorem 1] Let $B = \{B_i : D_i \to \{0, 1\} | i \in S_n, n \in N\}$ be a ΦF-unapproximable and υ-accessible set of predicates (where F is as defined earlier in relation to Γ) with a corresponding set of "friendly functions $f = \{f_i | i \in S_n, f_i : D_i \to D_i\}$, where $I = \{(i, x) | n \in \mathbb{N}, i \in S_n, x \in D_i\}$ is the set of all inputs relative to B. Then the following conditions are sufficient to ensure the existance (as constructed in the proof) of a $\Phi\Gamma\Upsilon - PRG$ (call it G).

1. f_i is a permutation on D_i (for all $i \in S_n$)
2. $f : (i, x) \in I \to D_i$ is on the order of some family of functions Υ
3. $h : (i, x) \in I \to B_i(f_i(x)) \in O(\Upsilon)$
4. $\upsilon \in O(\Upsilon)$.

Proof. Let $n \in \mathbb{N}$. Because B is υ-accessible, let c_0 and A be its relative constant and probabilistic algorithm. Choose some function $Q \in \Upsilon$ and set $c = Q(n)$, the desired length of the sequence, and $n' = \lfloor n^{\frac{1}{c_1}} \rfloor$. The following constitutes the $\Phi\Gamma\Upsilon - PRG$, G, which stretches a random n-bit seed r to a $Q(n)$-bit pseudo-random sequence.

[Generator G] Run A on r and define x_l as the l^{th} bit of r, for all $l \in \{1, \cdots, n\}$. If A's output is "?" then generate the sequence consisting of c 0's. Else A has selected an input (i, x) from I_n with uniform probability. Now, compute sequence $T_{(i,x)} = x, f_i(x), f_i^2(x), \cdots, f_i^c(x)$. Then, from left to right, extract one bit from each element in $T_{(i,x)}$ via $B_i(f_i^j(x))$, from $j = c$ to 1. (That is, reverse the sequence and apply the predicate to every element in $T_{(i,x)}$, thus attaining our desired $G(x_1, \cdots, x_n) = (y_1, \cdots, y_{Q(n)})$).

For simplicity, assume A never outputs "?" and $n = n'$. Thus G takes the random n-bit input r and stretches it into the sequence $s_1, \cdots, s_c$, where $s_j = B_i(f_i^{c-j+1}(x))$ (for $j \in 1, \cdots, c$).

First we prove G is computable in time on $O(\Upsilon)$. We assumed $f_p(x)$ (evaluating our binomial at a point) and $h(p, x)$ (applying the predicate of $f(x)$ given x) are both on $O(\Upsilon)$. Thus, given (p, x), G computes each s_j in time on $O(\Upsilon)$, since Υ is closed under addition. To get (p, x) from our given seed r requires time $\upsilon(n) \in \Upsilon$. Since the sum of three elements in Υ is still on $O(\Upsilon)$, we see G computes each s_j in time on $O(\Upsilon)$ from the given r.

Now we prove G is cryptographically secure (as per Definition 2.1). Let $X \in \Phi_1$ and $Z \in F$. We want to prove that, when n is large enough, for any $k \in \{1, \cdots, Q(n)\}$, any circuit C with $\leq Z(n)$ gates, when supplied with $s_1, \cdots, s_k$, cannot "predict" s_{k+1} with probability greater than $\frac{1}{2} + \frac{1}{X(n)}$ (over all n-bit inputs $(i, x) \in I_n$). This is even if the adversarial algorithm knows B, f, and h: as long as it does not know the seed. If proved, this would imply that neither could any algorithm in Γ predict s_{k+1} with such accuracy.

Assume, by contradiction, that there exists an infinite family $F \in \mathbb{N}$ such that for every $n \in F$, there exists a circuit C_n with less than $Z(n)$ gates which predicts some s_{k+1}, given from $s_1, \cdots, s_k$ for a fraction $\geq \frac{1}{2} + \frac{1}{X(n)}$ of all $(i, x) \in I_n$. Then the following algorithm $D \in \Gamma$, making calls to the circuits C_n, can $\frac{1}{X(n)}$-approximate B for inputs $(i, x) \in I_n$, where $n \in F$.

[Algorithm D] Given $(i, x) \in I_n$, where $n \in F$, generate the sequence $(b_1, \cdots, b_k) = (B_i(f_i^k(x)), \cdots, B_i(f_i(x)))$ as outlined in algorithm G. Input the first k bits to

the circuit C_n and set the output as s_{k+1}. Predict by $D(i,x) = s_{k+1}$.

Because C_n can $\frac{1}{X(n)}$-approximate B for $n \in F$, and because b_i is we see $D[x,i]$ correctly predicts s_{k+1} for a fraction at least $\frac{1}{2} + \frac{1}{X(n)}$ of the $(i,x) \in I_n$, and it runs in time on $O(F)$. This contradicts the ΦF-unapproximability of B.

□

4 Neccessary Conditions for a $\Phi\Gamma\Upsilon$-PRG

Observe that we need $\Upsilon \subseteq F$. For if, by contradiction, we had $F \subsetneq \Upsilon$, then there may be some function $g \in \Upsilon$ which can $\frac{1}{\phi(n)}$-approximate B for some $\phi \in \Phi$. Since G generates bits in time $O(\Upsilon)$, this means it could take longer to calculate bits of G than to $\frac{1}{\phi(n)}$-approximate B. In other words, $F \subsetneq \Gamma$ allows for the possibility of it taking less time for an adversary to break (predict) G than it does for the encrypter to even compute G.

By $\tau_f(n)$ and $\tau_h(n)$, we denote, respectively, the time it takes to calculate $f(i,x)$ and $h(i,x)$, using the fastest known algorithms, for any $(i,x) \in I_n$.

Let Δ denote the set of all functions on $O(\delta(n) + \tau_h(n))$, where $\delta(n)$ is the time it takes the fastest known algorithm to solve for x, given only $f_i(x)$, for all $(i,x) \in I_n$.

Consider what would happen if $\Delta \subseteq F$. Then, if some predicate B is ΦF-unapproximable (for some Φ), this means no algorithm in Δ (because $\Delta \subseteq F$) can solve $B_i(x)$ for a certain fraction of the inputs $(i,x) \in I_n$. For a given $(i,y) \in I_n$, we can find an $x \in D_i$ such that $f(i,x) = y$ in time $\delta(n)$, by assumption. So in time $\delta(n) + \tau_h(n)$, we can find this x and then take $B_i(f(i,x)) = B(y)$. That is, in time $\delta(n) + \tau_h(n) \in \Delta \in F$, we can solve for $B_i(y)$ given only y, for any $y \in D_i$: a contradiction of B being unapproximable by F. Thus if $\Delta \in F$, no predicate can be ΦF-unapproximable. To make a PRG, then, we require:

$$\Upsilon \subseteq F \subsetneq \Delta.$$

[Remark] If $\delta(n) \in O(n^2 \log(n))$, then f_p cannot be used to create a PRG (for any predicate) as outlined in the definition of Δ, since we require $\Upsilon \subseteq F \subsetneq \Delta$. That is, for this binomial to create a PRG, we need no algorithm running in time on $O(\log^2(p)\log(\log(p))$ to be able to solve for a root of $f(x) = x^a + cx^b - y \in \mathbb{F}_p^*[x]$. Trinomial solving in $O(\log^2(p)\log(\log(p)))$ time renders binomials useless for forming PRG's.

The following is the general strategy one must take to construct a $\Phi\Gamma\Upsilon$-PRG in the form of Blum and Micali (as outlined in Theorem 1).

1. Given an arbitrary n-bit integer, find the time it takes to generate an element $(i,x) \in (S_n, D_i)$ with uniform probability (over I_n). This gives $\phi(n)$.

2. Find the minimal times $\tau_f(n)$ and $\tau_h(n)$ it takes to calculate $f(i,x)$, and $h(i,x)$ respectively (and also D_i) for any $i \in S_n$.
3. Set Υ as all functions on $O(\tau_f(n) + \tau_h(n) + \phi(n))$.
4. Consulting current literature, find the fastest time it takes to solve for x given only $f(i,x)$ for all $(i,x) \in I_n$. Call this $\delta(n)$ and define Δ as the set of functions on $O(\delta(n) + \tau_h(n))$.
5. If $\Upsilon \subsetneq \Delta$, find Φ and F such that $\Upsilon \subseteq F \subsetneq \Delta$ and B is ΦF-unapproximable. This F will define Γ.

5 Binomial PRG's and Trinomial Hardness

We now consider how strong a certain binomial is in forming PRG's. Given current bounds on computation, what Φ, Γ, and Υ we can choose to fit the hypotheses on Theorem 1? We use the above outlined approach.

5.1 Binomial PRG

Definition 5.1. Let S_n be the set of n-bit primes. For each $p \in S_n$, choose $a, b, c \in \mathbb{F}_p^*$ such that $\gcd(a,b) = 1$ and $\gcd(a, p-1), \gcd(b, p-1) \geq 2$. Define the binomial friendly function

$$f_p := x^a + cx^b \in \mathbb{F}_p^*[x]$$

together with its inputs D_p as a subset of $\text{ran}(f_p)$ such that $|D_p| \geq Q(n)$ and D_p forms a cycle under f_p. That is, $\forall\ x, y \in D_p$, $\exists\ i \in \{0, \cdots, |D_p| - 1\}$ such that $y = f^{(i)}(x)$ (the i'th iterate of x under f_p). Then define the corresponding predicate $B_p : D_p \to \{0,1\}$ by

$$B_p(f(x)) = \begin{cases} 1 \text{ if } x \geq \text{median}(D_p) \\ 0 \text{ otherwise.} \end{cases}$$

The resulting PRG (formed via Theorem 1) is the Binomial PRG.

The most computationally expensive aspect of this friendly function is that we must, given a prime p, find such an $(a,b,c) \in (\mathbb{F}_p^*)^3$ and D_p where f_p is a permutation on D_p and $|D_p| \geq Q(n)$.

Recall that $Q(n) \geq n$. If $|D_i| < n$, then our PRG sequence (for an n-bit x) is $(y_1, \cdots, y_{D_i}, \cdots, y_{Q(n)})$, which is (α, β)-periodic where $\alpha + \beta \geq |D_i|$. As mentioned earlier, and in Blum and Micali's paper, the algorithm which predicts the next bit of this PRG sequence by guessing either randomly or in order to preserve any existing periodicity will predict next-bits with troublesome accuracy for all such n. Thus we require that $Q(n) \geq n$ and that $|D_i| \geq n$ (past some n_0).

One choice which has been studied already is the special form $f_p(x) = cx^d + x$. The simplest case of this is when f is a bijection on $\mathbb{F}_p^*$; that is, by [Kai Lu], when $d = \frac{p+1}{2}$ and $1 - c^2$ is a square. Observe, however, that $x^d = x^{\frac{p-1}{2}+1} = \pm x$. Thus $f_p(x) = cx^d + x = (1 \pm c)x$, so given a $y \in \mathbb{F}_p^*$, to find the $x \in \mathbb{F}_p^*$ such that $f(x) = y$, simply test $x = y(1 \pm c)^{-1}$. Thus this d cannot be used for our purposes as a friendly function, regardless of c.

By previous REU students, it is known that the number of roots of f_p is $\leq \lfloor \sqrt{p-1}$, and it is conjectured (based on experimental data) that p being prime means the number of roots of f_p is on $O(n)$, where p is n-bits. This bounds the discrepency of f_p over F_p^*, thereby also bounding $ran(f_p)$.

5.2 Accessibility of B

Given an n-bit integer, how long does it take to uniformly choose an input (p, x) to our binomial, and how long does computing $f(p, x)$ and $h(p, x)$ take? This requires finding an n-bit prime p, an $x \in \mathbb{F}_p^*$, and a, b, c, and D_p such that $x^a + cx^b$ is a permutation on D_p? This must be doable faster than $2^{\frac{n}{2}} \log(n)$.

To begin, note that $f_p : x \in D_p \to D_p$ is on $O(n^2 \log(n))$ according to current computation bounds (December 2021) [RZ21].

Now, given an n-bit input r, how much time does it take to uniformly choose an n-bit prime p? By Solovay and Strassen's work on Monte-Carlo primality tests [SS77], we can check if r is prime (with 0 chance of false positive) on time on $O(\log(n))$ (to be precise, $6 \log_2(n)$). The chance of a false negative, when the Solovay-Strassen Primality test is applied m times, is 2^{-m}. By the Frequency of Primes Theory, the expected number of random samplings (among n-bit integers) before picking a prime is on $O(n)$. So if we do $k \in O(n)$ many random samplings of r and perform the SSP test $m = 1,000$ times for each (to make the finer details of the statistics negligable), we see that it takes time $O(n^2 \log(n))$ to choose such a p, with the result uniform over n-bit primes. Thus the time it takes to uniformly select a $(p, x) \in I_n$ is on $O(n + n^2 \log(n))$, or, simply, $O(n^2 \log(n))$.

A necessary condition for a $\Phi\Gamma\Upsilon$-PRG is that $\delta \notin O(\Gamma)$. The best known $\delta(n)$ is 2^n, and Υ (which requires time to compute f_p given a, b, c, and D_p) is at least $n^2 \log(n)$. Hence we need to be able to find a, b, c, and D_p (such that $x^a + cx^b$ is a permutation on D_p) in time on $O(2^{n/2} \log(n)) = O(p \log(\log(p)))$. Finding this D_p systematically is the primary difficulty of this PRG.

One method* is, for each p, to pick an arbitrary $a, b, c, x \in \mathbb{F}_p^*$ such that $\gcd(a, p-1), \gcd(b, p-1) \geq 2$, $\gcd(a, b) = 1$, and c is prime. Then, iterate x under $x^a + cx^b$ until finding $i < j$ such that $f_p^{(i)}(x) = f_p^{(j)}(x)$. Then $j - i$ (the cycle length) must be in $(Q(n), \frac{2^n}{-}]$, so that $|D_p| \geq Q(n)$ and iterating through $f_p(x)$ (doable on $O(\Upsilon)$) takes time significantly less than on $O(\delta(n))$ (hence the added n in the denominator of the upper bound). We also need $i \in [0,]$. These bounds demand most seeds to quickly map to a cycle of proper length.

For simplicity, assume $Q(n) = n$ and $\Phi = POLY$. For this method to efficiently find us an a, b, c, and D_p, we need a fraction $> 1 - \frac{1}{n}$ of all $x \in \mathbb{F}_p^*$ to fit the above bounds.

5.3 Experimental Results

We will begin this section with a comparison of three equations over the finitie field F_p with $p = 1009$. These experiments highlight three things:

1. Some a, b, c result in predictable behavior over the field, making them easy to approximate.
2. Even these bad a, b, c can have discrete Fourier transforms that seem random (look like white noise); hence discrete Fourier Analysis is not sufficient to identify Pseudo-Random Behavior.
3. Suitable cycles are possible with certain a, b, c resulting in valid D_p.

The following figures contain, respectively,an iteration sequence plot over an arbitrary input with the length of the sequence being p, the output of the function itself for each element in F_p, a Discrete Fourier spectrum of that iteration, and a functional graph of the equation over the finite field. Notice that each output plot and iteration plot map from F_p^* to F_p^*. Finally, the functional graphs are directed graphs where each node is an element of F_p^* and each edge shows where the element maps to under f_p. A close up is given to illustrate particular behavior.

The exponential $f(x) = 11^x$ in figure 1 is included as a first example as much of our work compares to the DLP-PRG. Notice the cycles in the iteration graph. Additionally the low visual discrepancy shown in the iteration plot is not picked up by the discrete Fourier analysis. In other words, the seeming random behavior despite cycling seemingly fools classical discrete Fourier analysis. Additionally, as will be seen, the number of components of the functional graph is relatively small. This means it it seems easier to get into a cycle in the exponential case. This result is expanded below.

Despite being quickly shown to not be a good choice for c, d, figure 2 of $f(x) = x + cx^{(p+1)/2}$ is interesting. It illustrates graphically, linear behavior plus noise over the whole field, which first caused us to realize the frailty of that choices security. Despite this, the iteration behaviour looks fairly random. Similar to the exponential case, the discrete Fourier analysis is not able to identify this randomness and has output similar to white noise. Finally, even in this permutation scenario, the number of components is higher than the exponential case, with many small cycles.

Figure 3 of $f(x) = x^7 + 606x^{505}$ shows what we have found to be the most common behavior. Typically, there is fairly random behavior over the whole field, but, small cycles are quickly found. Indeed, the functional graph has 936 components in this example, which suggests that suitable large cycles don't exist

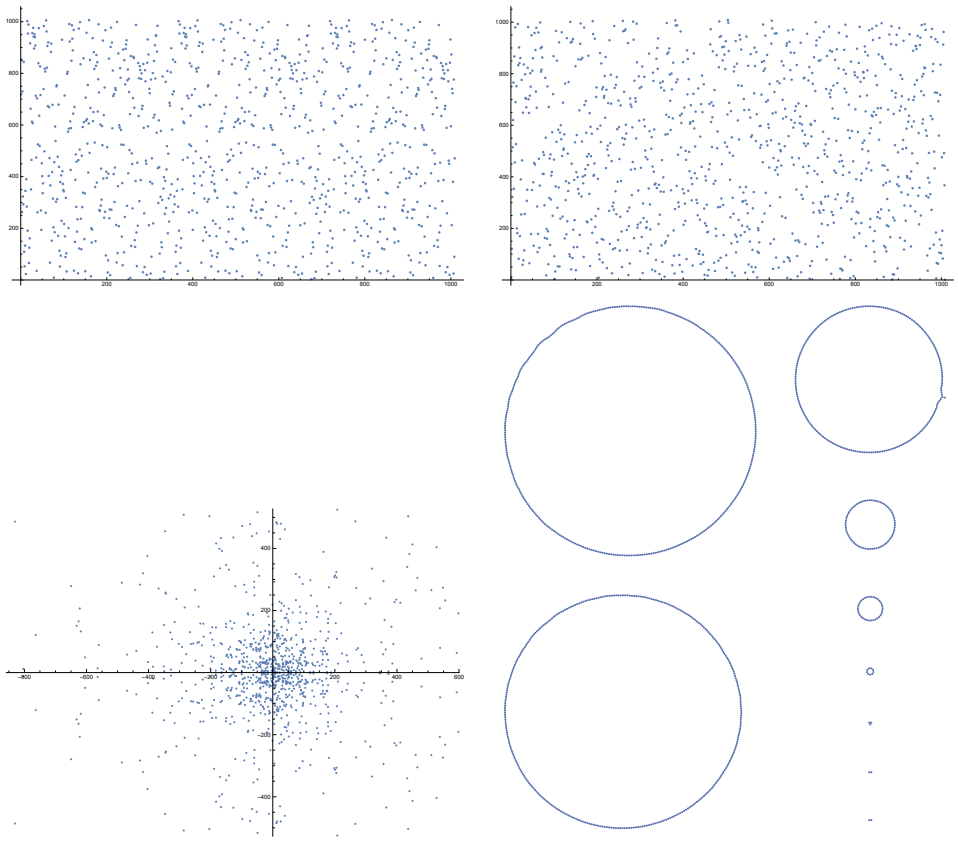

Figure 1: $f(x) = 11^x \mod 1009$, $p = 1009$, Itervalue (range of values under itaration) : 582(top left), Number of Components (in functional graph): 10

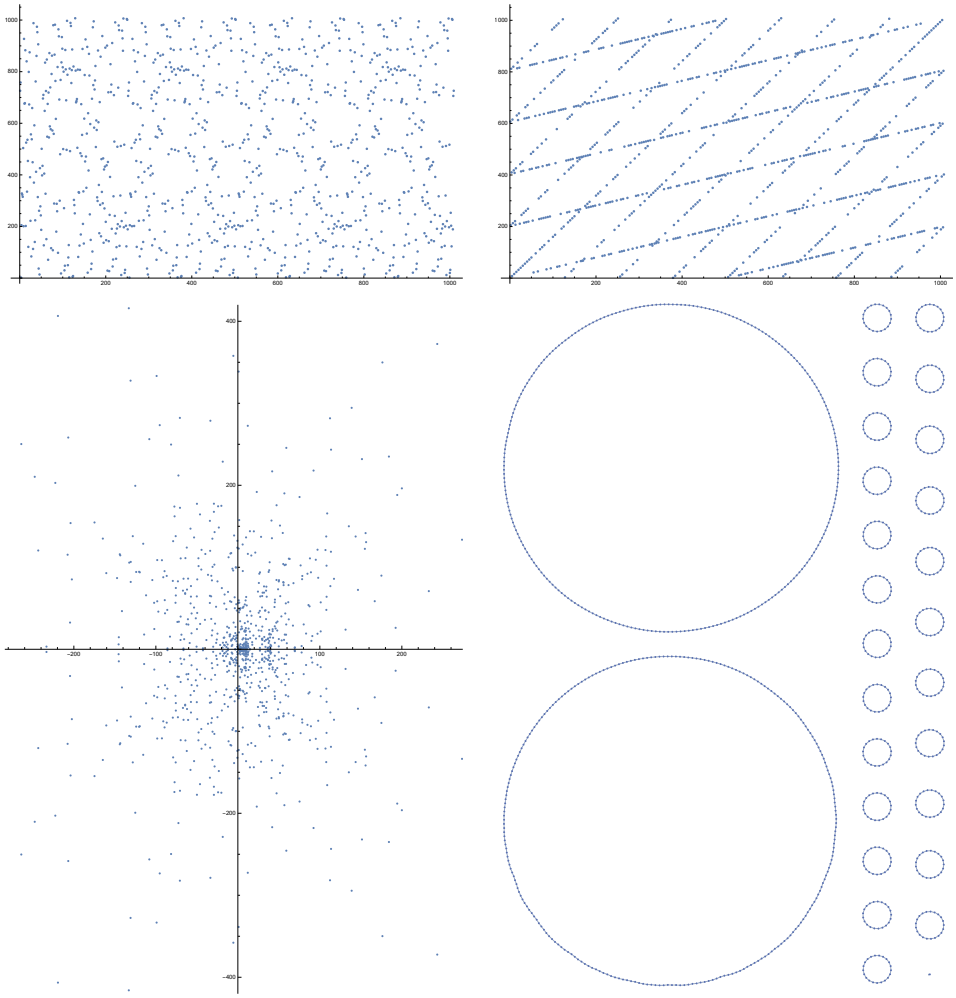

Figure 2: $f(x) = x + cx^{(p+1)/2}$, $p = 1009$, Itervalue: 706(top left), $c = 606$ satisfies $1 - c^2 = d^2$ where $d \in F_p$, Number of Components: 27

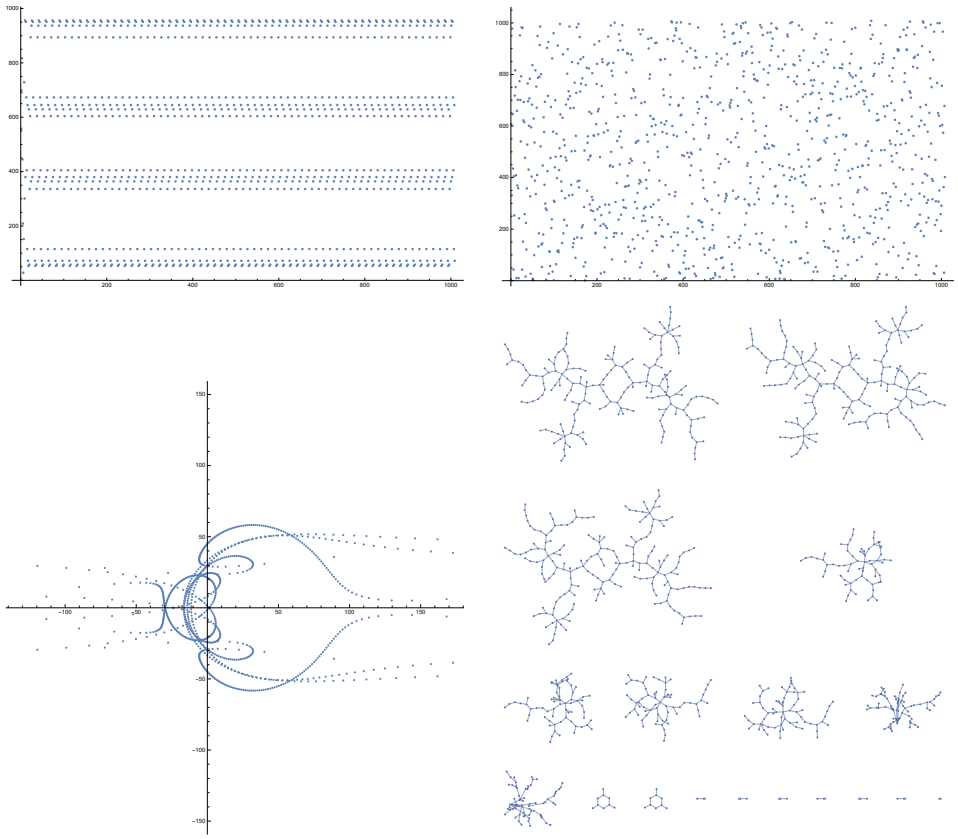

Figure 3: $f(x) = x^7 + 606x^{505}$, $p = 1009$, Itervalue: 756 (top left), Components: 936

in this choice of a, b, c. Additionally, the shortness of the cycle chosen is able to be picked up by the discrete Fourier analysis, with some very clear structure being shown. There seems to be some bound relating discrepancy to the ability of discrete Fourier analysis to identify pseudo-random behavior. This is not a formalized result, but points in that direction.

Figure 4 is important because it give us hope that suitable a, b, c can be found. Notice the random seeming behavior over both the iteration as well as the field. Similarly, there are large cycles in the graph. However, there are a number of singular points that cannot easily be visualised given the scale of the graph. In common with our theme so far, the Fourier analysis is not sufficient in identifying pseudo-random behavior with low enough discrepancy.

Figure 5 corresponds to a close up to one of the large cycles in figure 4. This figure should also make more clear what exactly is going on in the functional graphs for all of the examples hitherto mentioned.

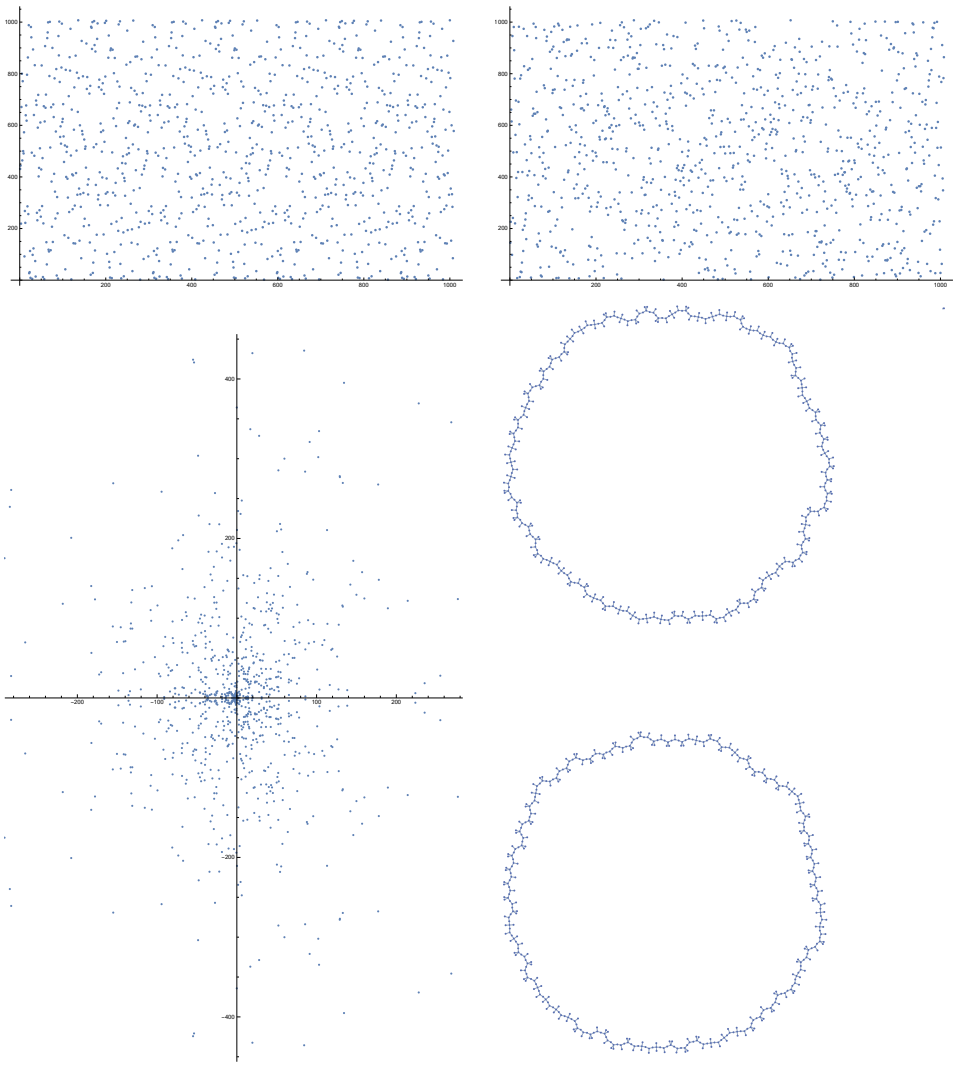

Figure 4: $f(x) = x^7 + 144x^{151}$, $p = 1009$, Itervalue: 82 (top left), Components: 435

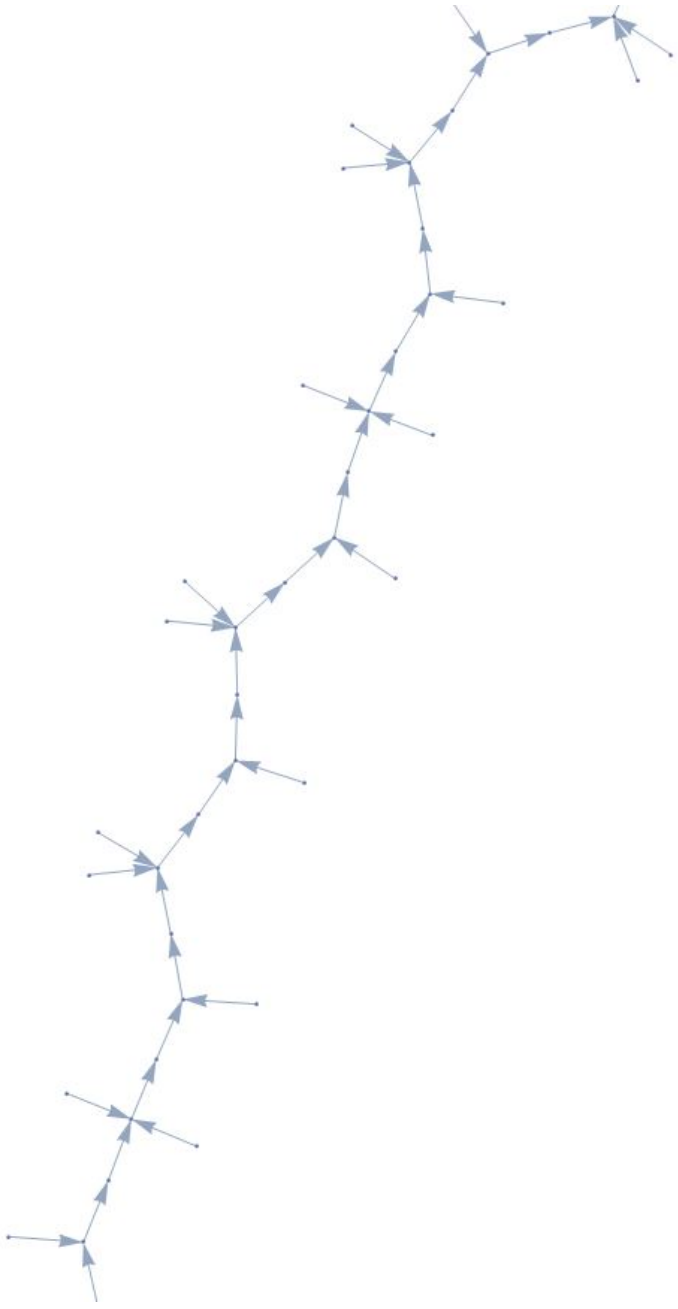

Figure 5: Closeup of Section of a Cycle in a Functional Graph

What we notice is that this binomial seems to create rather small cycles, though it does so quickly, for arbitrary seeds. The relatively large number of connected components also speaks to this.

While these experimental results give us an idea of behavior, we want more generalized information. Figure 6 corresponds to the fraction of $a = 1, c, d$ such that enough (a fraction $> 1 - \frac{1}{poly(n)}$) of the seeds that 'quickly' map to a cycle of 'proper length' (as defined earlier). Figure 7 is the same, except over all a, b, c, and x choices.

What is important to note, is the seeming exponential decay. This implies that for a given $Q(n)$ we want p sufficiently greater than $Q(n)$ to be able to generate a c, d for our seed x that will allow us to generate c, d in $O(\upsilon)$ time. Figure 8 corresponds to this same result but for choices of g, x. It follows that there could be some established relationship between the two decays exhibited. Regardless, it seems that it might be easier to find a suitable cycle on F_p^* for the exponential case than it is for the polynomial case. This could mean that finding a, b, c is prohibitively expensive, but, without formal results this is not concrete.

5.4 Unapproximability of B

The following two assumptions are necessary for proving unapproximability, and how strong of a $w(n)$ can be chosen determines the strength of the PRG, as will be seen.

Assumption 1 (Predicate-to-Roots Assumption (PRA))**.** There exists an algorithm running in time on $O(w(n))$ such that, if for a fraction $> 1 - \frac{1}{P(n)}$ of all $y \in D_p$, we know $B_p(y)$, then our algorithm outputs an $x \in D_p$ satisfying $x^a + cx^b = y$.

Assumption 2 (Trinomial Hardness Assumption (THA))**.** No algorithm running in time on $O(w(n))$ can solve for the root of $f_p(x) - y$ for a fraction $> \frac{1}{2} + \frac{1}{P(n)}$ of the $y \in \mathbb{F}_p^*$, for any polynomial P and n-bit prime p.

Lemma 1. Then the binomial predicate B is $\Phi\Gamma$-unapproximable (for $\Phi = POLY$) if all algorithms in Γ run in time on $O(w(n))$, and THA and PRA are satisfied by $w(n)$.

Proof. Assume the THA and the PRA for some fixed $w(n)$, and let X be the family of circuits assumed to exist by the PRA. Suppose, by contradiction, B is not $\Phi\Gamma$-unapproximable. That is, there exists infinite set $T \subset \mathbb{N}$ such that for every $n \in T$, there exists a circuit C_n (with number of gates on $O(F)$) correctly computing $B_p(x)$ for a fraction $\geq \frac{1}{2} + \frac{1}{\phi(n)}$ of all $(p, x) \in I_n$, for some $\phi \in \Phi$. By a counting argument [Blum-Micali], for all $n \in T$, a fraction $> \frac{1}{\phi(n)}$ of all n-bit primes p have this property: that $C_n[p, x] = B_p(x)$ for a fraction $> \frac{1}{2} + \frac{1}{2\phi(n)}$ of all $x \in \mathbb{F}_p^*$. Then there exists an algorithm Y, making calls

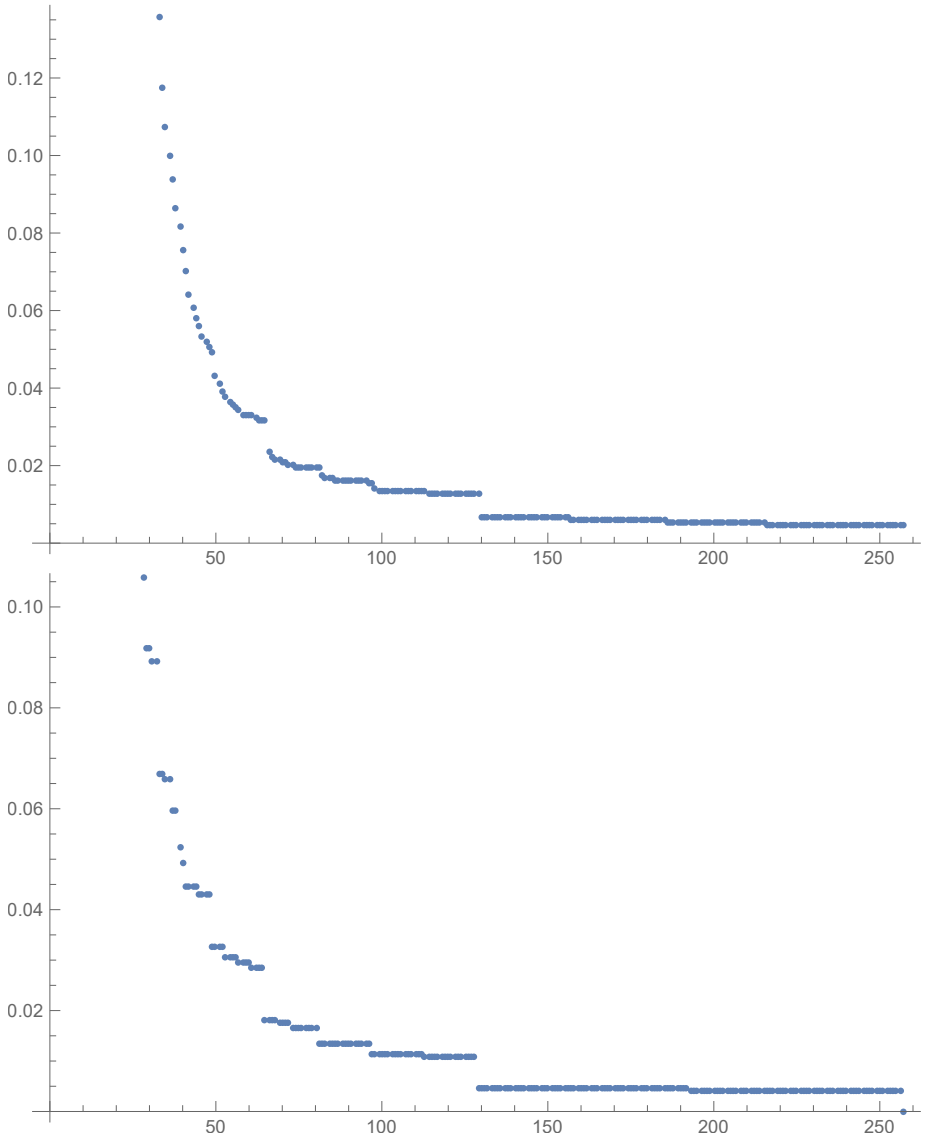

Figure 6: Fraction of c, d, x(Y Axis) for $f(x) = x + cx^d \mod p$ on F_p, $p = 1009$, with valid Pre-Cycle plus Cycle Lengths (Top Graph X Axis)(Top), and only Cycle Satisfying Certain Length(Bottom Graph X Axis)(Bottom)

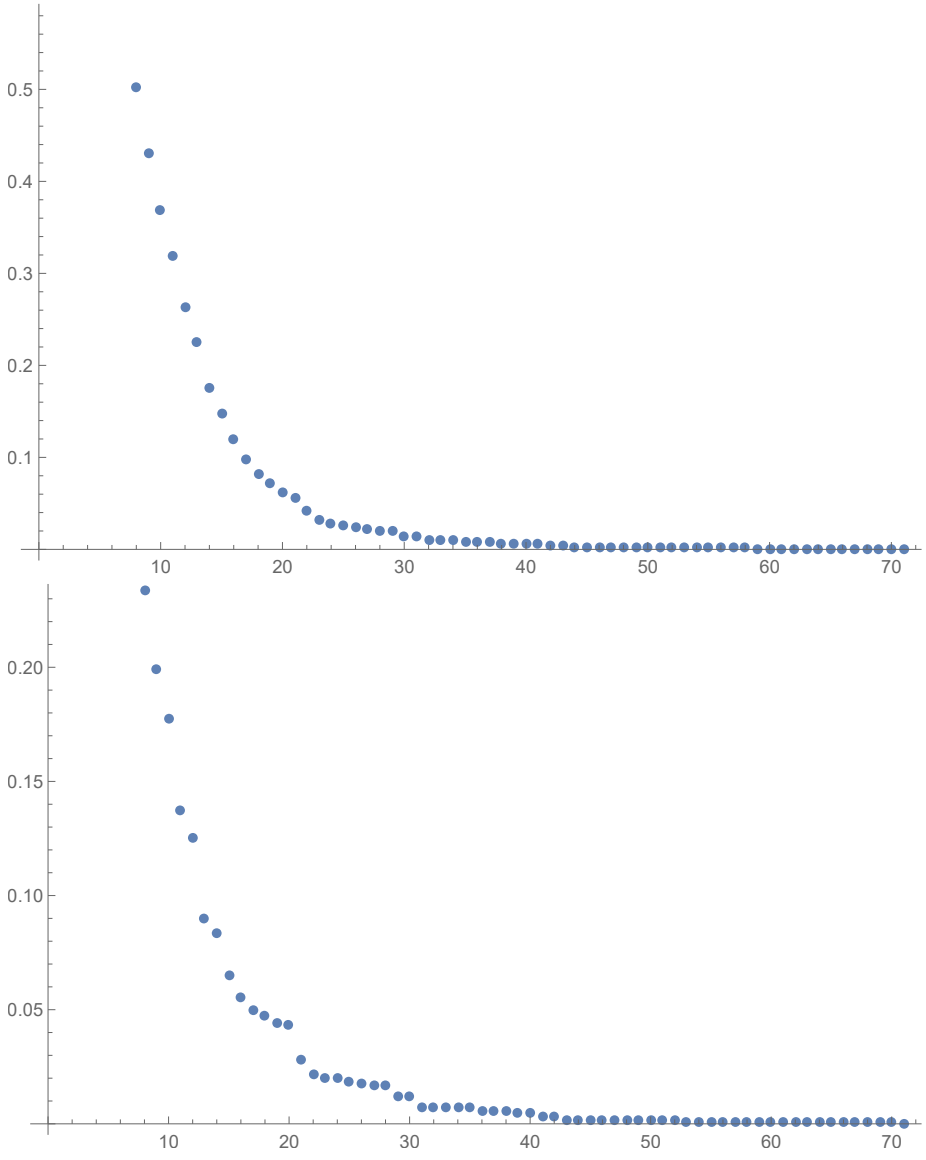

Figure 7: Fraction of a, c, d, x(Y Axis) for $f(x) = x^a + cx^d \mod p$ on F_p with $p = 71$ with Pre-Cycle plus Cycle Satisfying Certain Length (X Axis)(Top), and only Cycle Satisfying Certain Length(X Axis)(Bottom)

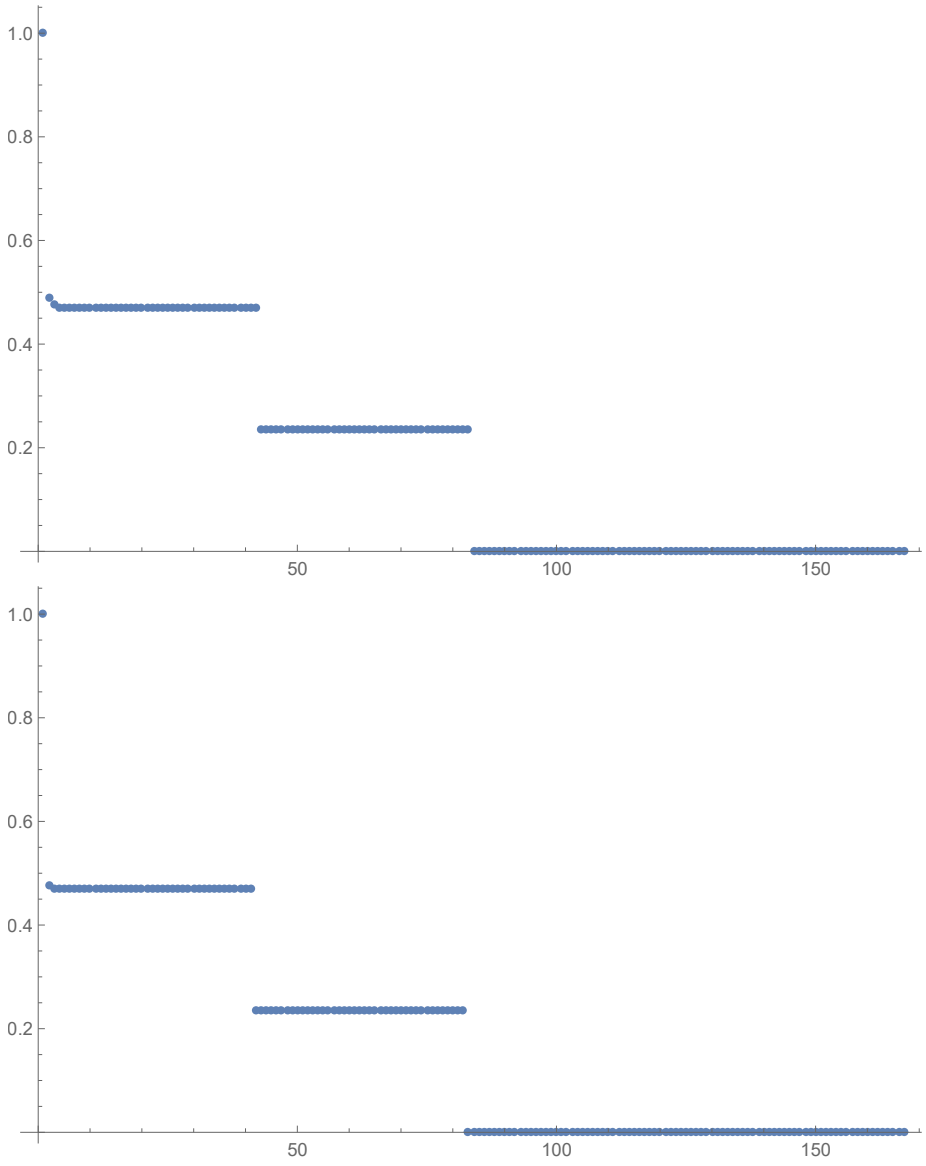

Figure 8: Fraction of g, x(Y Axis) for $f(x) = g^x \mod p$ where g ensures a permutation on F_p with $p = 167$ with Pre-Cycle plus Cycle Satisfying Certain Length (X Axis)(Top), and only Cycle Satisfying Certain Length(X Axis)(Bottom)

to C_n and X as such: for any n-bit prime p, when given $a \in \mathbb{F}_p^*$, Y guesses $B_p(y) = C_n[p, y]$. The algorithm X, assuming this output to be in fact the predicate $B_p(y)$, computes the $x \in \mathbb{F}_p^*$ such that $x^a + cx^b = y \pmod p$, and thus Y guesses this x as the root of f_p. Now Y can be transformed (via a transformation on X, allowing it to make calls on C_n) into a circuit with a number of gates on $O(w(n)) + O(F)$, which means on $O(w(n))$ (because of the sizes of C_n and X.) Also, by construction, for all $n \in T$, Y correctly predicts the root of $x^a + cx^b - y$ for a fraction $> \frac{1}{2} + \frac{1}{2\phi(n)}$ of all $y \in \mathbb{F}_p^*$. This contradicts the THA, using this proof's ϕ, Φ, and $w(n)$, and thus B is $\Phi\Gamma$-unapproximable. $\square$

6 Comparing Binomials and DLP PRG's

Implicit in Blum and Miculi's Discrete Logarithm PRG (DLP-PRG) is the use of a specific Γ, Φ, and Υ. Their friendly function is $f(x) = g^x \pmod p$, their seed i is a concatenation of a prime p and a generator g or $\mathbb{F}_p^*$, and their predicate is defined as such:

$$B_i(x) = \begin{cases} 1 \text{ if } x \text{ is the principal square root of } x^2 \mod p \\ 0 \text{ otherwise.} \end{cases}$$

Blum and Micali use the set of all polynomials on n as their F, Φ, and Υ for simplicity, this size coming from their assumption of the hardness of the DLP, though there are known algorithms on $O(2^{2\sqrt{\log p \log\log p}})$ that solve DLP, and thus their bounds may be different. The Binomial PRG has a smaller Γ and Υ, meaning (if it works) it is easier to generate and easier to break.

7 Observations about PRG's

[Theorem 2] If f is a friendly function for a $\Phi\Gamma\Upsilon$-PRG, then there does not exist a predicate such that f^{-1}, defined with the same set of inputs as f, generates a $\Phi\Gamma\Upsilon$-PRG (for the same Γ,Φ, and Υ).

Proof. When we speak of a $\Phi\Gamma\Upsilon$-PRG, we mean one specifically generated in the way outlined in Theorem 1. Let f be a friendly function forming a $\Phi\Gamma\Upsilon$-PRG with predicate $B^{(1)}$ and corresponding $h^{(1)}(x) = B^{(1)}(f(x))$. Let the set of inputs of $f = \{f_i | i \in S_n\}$ be $I = \{(i, x) | x \in D_i, i \in S_n, n \in \mathbb{N}\}$. Let the set of inputs for $f^{-1} = \{f_i^{-1} | i \in S_n\}$ also be I, where $f_i^{-1} : D_i \to D_i$ is defined on each i, and for every $x, y \in D_i$, by $f_i(x) = y \iff f_i^{-1}(y) = x$. Assume, by contradiction, that there exists a predicate $B^{(2)}$ allowed f^{-1} to also form a $\Phi\Gamma\Upsilon$-PRG, with a $h^{(2)}(y) = B^{(2)}(f^{-1}(y))$. Note that $B^{(1)}$ is $\Phi\Gamma$-unapproximable, and, as noted earlier, $\Upsilon \subsetneq F$ (the F determining Γ). Notice that f, f^{-1}, $h^{(1)}$, and $h^{(2)}$ are all on $O(\Upsilon)$. Let some i and $y \in D_i$ be given. We can choose $x \in D_i$ such that $f_i(x) = y$ and $f_i^{-1}(y) = x$. Since f_i^{-1} runs in at worst time on $O(\Upsilon)$, the time it takes to find such an x is on $O(\Upsilon)$. Then $h_i^{(1)}(x) = h_i^{(1)}(f_i^{-1}(y)) = B_i^{(1)}(f_i(f_i^{-1}(y))) = B_i^{(1)}(y)$, because by Theorem

1 assumptions f is injective, and this operation we said can be done on at worst time on $O(\Upsilon)$. Since Υ is closed under addition, putting the previous two steps together, we see that deterministically (that is, with 100-percent accuracy) finding $B_i^{(1)}(y)$ from any $y \in D_i$ is doable in time on $O(\Upsilon)$ (of length n of i) for any $i \in S_n$. Thus there exists function n $O(\Upsilon)$ solving for the predicate, meaning Γ cannot be contained in Υ, thus $\Gamma \subsetneq \Upsilon$, a contradiction. $\square$

8 Future Directions

Do Gowers Norms or Higher Order Fourier Analysis give more promising results to identify pseudo-random sequence behavior?

Can one formalize the experimental results to allow their use in formal proof? Further, running the above method* (page 7) on each n-bit prime p for varying n, we could find how expensive finding D_p is.

Does there exist a suitable predicate B that deeply relates the roots of trinomials to the distribution over D_p such that the PRA and THA can be used to prove unapproximability? One avenue we would like to look into is the relationship between discrepancy and the number of roots of a trinomial.

Idea: A given function f has a matching predicate B to form a PRG if f^{-1} cannot be computed on the order of the time it takes to evalute f.

References

[1] [BM84] Manuel Blum and Silvio Micali, "How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits," SIAM J. Comput., Vol. 13, No. 4, Nov. 1984.

[2] [SS77] Solovay, Robert M.; Strassen, Volker (1977). "A fast Monte-Carlo test for primality". SIAM Journal on Computing.

[3] [RZ21] J. Maurice Rojas and Yuyu Zhu (2021), "Root Repulsion and Faster Solving for Very Sparse Polynomials over p-Adic Fields". ISSAC 2021 Proceedings.