Close to Uniform Prime Number Generation With Fewer Random Bits

Pierre-Alain Fouque[1] and Mehdi Tibouchi[2]

1 **Université Rennes 1 and Institut Universitaire de France,** Pierre-Alain.Fouque@ens.fr
2 **NTT Secure Platform Laboratories – 3–9–11 Midori-cho, Musashino-shi, Tokyo 180–8585, Japan** tibouchi.mehdi@lab.ntt.co.jp

Abstract

In this paper, we analyze several variants of a simple method for generating prime numbers with fewer random bits. To generate a prime p less than x, the basic idea is to fix a constant $q \propto x^{1-\varepsilon}$, pick a uniformly random $a < q$ coprime to q, and choose p of the form $a + t \cdot q$, where only t is updated if the primality test fails. We prove that variants of this approach provide prime generation algorithms requiring few random bits and whose output distribution is close to uniform, under less and less expensive assumptions: first a relatively strong conjecture by H. Montgomery, made precise by Friedlander and Granville; then the Extended Riemann Hypothesis; and finally fully unconditionally using the Barban–Davenport–Halberstam theorem.

We argue that this approach has a number of desirable properties compared to previous algorithms. In particular:

- it uses much fewer random bits than both the "trivial algorithm" (testing random numbers less than x for primality) and Maurer's almost uniform prime generation algorithm;
- the distance of its output distribution to uniform can be made arbitrarily small, unlike algorithms like PRIMEINC (studied by Brandt and Damgård), which we show exhibit significant biases;
- all quality measures (number of primality tests, output entropy, randomness, etc.) can be obtained under very standard conjectures or even unconditionally, whereas most previous nontrivial algorithms can only be proved based on stronger, less standard assumptions like the Hardy–Littlewood prime tuple conjecture.

1998 ACM Subject Classification G.4 Algorithm design and analysis

Digital Object Identifier 10.4230/LIPIcs.xxx.yyy.p
Keywords: Number Theory, Cryptography, Prime Number Generation.

Conference title on which this volume is based on.
Editors: Billy Editor, Bill Editors; pp. 1–19

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

There are several ways in which we could assess the quality of a random prime generation algorithm, such as its speed (time complexity), its accuracy (the probability that it outputs numbers that are in fact composite), its statistical properties (the regularity of the output distribution), and the number of bits of randomness it consumes to produce a prime number (as good randomness is crucial to key generation and not easy to come by [9]).

In a number of works in the literature, cryptographers have proposed faster prime generation algorithms [4, 3, 16, 15] or algorithms providing a proof that the generated numbers are indeed prime numbers [18, 19, 20].

A number of these works also prove lower bounds on the entropy of the distribution of prime numbers they generate, usual based on very strong conjectures on the regularity of prime numbers, such as the prime r-tuple conjecture of Hardy-Littlewood [13]. However, such bounds on the entropy do not ensure that the resulting distribution is statistically close to the uniform distribution: for example, they do not preclude the existence of efficient distinguishers from the uniform distribution, which can indeed be shown to exist in most cases.

But some cryptographic protocols (including most schemes based on the Strong RSA assumption, such as Cramer-Shoup signatures [5]) specifically require uniformly distributed prime numbers for the security proofs to go through[1].

Moreover, some cryptographers, like Maurer [18], have argued that even for more common uses of prime number generation, like RSA key generation, one should preferably generate primes that are almost uniform, so as to avoid biases in the RSA moduli N themselves, even if it is not immediately clear how such biases can help an adversary trying to factor N. This view is counterbalanced by results of Mihăilescu [21] stating in particular that, provided the biases are not too large (a condition that is satisfied by the algorithms with large output entropy mentioned above, if the conjectures used to establish those entropy bounds hold), then, asymptotically, they can give at most a polynomial advantage to an adversary trying to factor N. This makes the problem of uniformity in prime number generation somewhat comparable to the problem of tightness in security reductions.

To the authors' knowledge, the only known prime generation algorithms for which the statistical distance to the uniform distribution can be bounded are the one proposed by Maurer [18, 19] on the one hand, and the trivial algorithm (viz. pick a random odd integer in the desired interval, return it if it is prime, and try again otherwise) on the other hand. The output distribution of the trivial algorithm is exactly uniform (or at least statistically close, once one accounts for the compositeness probability of the underlying randomized primality checking algorithm), and the same can be said for at least some variants of Maurer's algorithm, but both of those algorithms have the drawback of consuming a very large amount of random bits.

By contrast, the PRIMEINC algorithm studied by Brandt and Damgård [3] (basically, pick a random number and increase it until a prime is found) only consumes roughly as many random bits as the size of the output primes, but we can show that its output distribution, even if it can be shown to have high entropy if the prime r-tuple conjecture holds, is also provably quite far from uniform, as we demonstrate in §4.1. It is likely that most algorithms

[1] In fact, it is sometimes enough to prove that *collision probability* is small. However, the authors are not aware of any prime number generator for which collision probability can be bounded other than those ensuring the uniformity of the output distribution.

that proceed deterministically beyond an initial random choice, including those of Joye, Paillier and Vaudenay [16, 15], exhibit similar distributional biases.

The goal of this paper is to achieve in some sense the best of both worlds: construct a prime generation algorithm that consumes much fewer random bits than the trivial algorithm while being efficient and having an output distribution that is provably close to the uniform one.

We present such an algorithm in §3: to generate a prime p, the basic idea is to fix a constant $q \sim x^{1-\varepsilon}$, pick a uniformly random $a < q$ coprime to q, and choose p of the form $a + t \cdot q$, where only t is updated if the primality test fails. We prove that variants of this approach provide prime generation algorithms requiring few random bits and whose output distribution is close to uniform, under less and less expensive assumptions: first a relatively strong conjecture by H. L. Montgomery, made precise by Friedlander and Granville; then the Extended Riemann Hypothesis; and finally fully unconditionally using the Barban–Davenport–Halberstam theorem.

2 Preliminaries

2.1 Regularity measures of finite probability distributions

In this subsection, we give some definitions on distances between random variables and the uniform distribution on a finite set. We also provide some relations which will be useful to bound the entropy of our prime generation algorithms. These results can be found in [24].

[Entropy and Statistical Distance] Let X and Y be two random variables on a finite set S. The *statistical distance* between them is defined as the ℓ_1 norm:[2]

$$\Delta_1(X;Y) = \sum_{s\in S} \Big| \Pr[X = s] - \Pr[Y = s] \Big|.$$

We simply denote by $\Delta_1(X)$ the statistical distance between X and the uniform distribution on S:

$$\Delta_1(X) = \sum_{s\in S} \Big| \Pr[X = s] - \frac{1}{|S|} \Big|,$$

and say that X is *statistically close to uniform* when $\Delta_1(X)$ is negligible.[3]

The *squared Euclidean imbalance* of X is the square of the ℓ_2 norm between X and the uniform distribution on the same set:

$$\Delta_2^2(X) = \sum_{s\in S} \Big| \Pr[X = s] - 1/|S| \Big|^2.$$

We also define the *collision probability* of X as:

$$\beta(X) = \sum_{s\in S} \Pr[X = s]^2,$$

and the *collision entropy* (also known as the Rényi entropy) of X is then $H_2(X) = -\log_2 \beta(X)$. Finally, the *min-entropy* of X is $H_\infty(X) = -\log_2 \gamma(X)$, where $\gamma(X) = \max_{s\in S}(\Pr[X = s])$.

---

[2] An alternate definition frequently found in the literature differs from this one by a constant factor 1/2. That constant factor is irrelevant for our purposes.

[3] For this to be well-defined, we of course need a family of random variables on increasingly large sets S. Usual abuses of language apply.

▶ **Lemma A.** *Suppose X is a random variable of a finite set S. The quantities defined above satisfy the following relations:*

$$\gamma(X)^2 \leq \beta(X) = 1/|S| + \Delta_2^2(X) \leq \gamma(X) \leq 1/|S| + \Delta_1(X), \tag{1}$$

$$\Delta_1(X) \leq \Delta_2(X)\sqrt{|S|}. \tag{2}$$

2.2 Prime numbers in arithmetic progressions

All algorithms proposed in this paper are based on the key idea that, for any given integer $q > 1$, prime numbers are essentially equidistributed among invertible classes modulo q. The first formalization of that idea is de la Vallée Poussin's *prime number theorem for arithmetic progressions* [8], which states that for any fixed $q > 1$ and any a coprime to q, the number $\pi(x; q, a)$ of prime numbers $p \leq x$ such that $p \equiv a \pmod q$ satisfies:

$$\pi(x; q, a) \underset{x\to+\infty}{\sim} \frac{\pi(x)}{\varphi(q)}. \tag{3}$$

De la Vallée Poussin established that estimate for constant q, but it is believed to hold uniformly in a very large range for q. In fact, H. L. Montgomery conjectured [22, 23] that for any $\varepsilon > 0$:[4]

$$\left|\pi(x; q, a) - \frac{\pi(x)}{\varphi(q)}\right| \ll_\varepsilon (x/q)^{1/2+\varepsilon} \qquad \big(q < x,\ (a, q) = 1\big),$$

which would imply that (3) holds uniformly for $q \ll x/\log^{2+\varepsilon} x$. However, Friedlander and Granville showed [10] that conjecture to be overly optimistic, and proposed the following corrected estimate.

▶ **Conjecture B** (Friedlander–Granville–Montgomery)**.** *For $q < x$, $(a, q) = 1$ and all $\varepsilon > 0$, we have:*

$$\left|\pi(x; q, a) - \frac{\pi(x)}{\varphi(q)}\right| \ll_\varepsilon (x/q)^{1/2} \cdot x^\varepsilon.$$

In particular, the estimate (3) *holds uniformly for $q \ll x^{1-3\varepsilon}$.*

That conjecture is much more precise than what can be proved using current techniques, however. The best unconditional result of the same form is the Siegel–Walfisz theorem [26], which only implies that (3) holds in the much smaller range $q \ll (\log x)^A$ (for any $A > 0$).

Stronger estimates can be established assuming the Extended Riemann Hypothesis (i.e. the Riemann Hypothesis for L-functions of Dirichlet characters/cyclotomic number fields), which gives [6, p. 125]:

$$\left|\pi(x; q, a) - \frac{\pi(x)}{\varphi(q)}\right| \ll x^{1/2} \log x \qquad \big(q < x,\ (a, q) = 1\big).$$

This implies (3) in the range $q \ll x^{1/2}/\log^{2+\varepsilon} x$, which is again much smaller than the one from Conjecture B. The range can be extended using averaging, however. The previous result under ERH is actually deduced from estimates on the character sums $\pi(x, \chi) = \sum_{p\leq x} \chi(p)$ for nontrivial Dirichlet characters $\chi \bmod q$, and more careful character sum arguments allowed Turán to obtain the following theorem.

---

[4] As is usual in analytic number theory and related subjects, we use the notations $f(u) \ll g(u)$ and $f(u) = O\big(g(u)\big)$ interchangeably. A subscripted variable on $\ll$ or O means that the implied constant depends only on that variable.

▶ **Theorem C** (Turán [25]). *The Extended Riemann Hypothesis implies that for all $q < x$:*

$$\sum_{a\in(\mathbb{Z}/q\mathbb{Z})^*} \left|\pi(x;q,a) - \frac{\pi(x)}{\varphi(q)}\right|^2 \ll x(\log x)^2$$

where the implied constant is absolute.

That estimate is nontrivial in the large range $q \ll x/\log^{4+\varepsilon}$, and implies that (3) holds for all q in that range and *almost all* $a \in (\mathbb{Z}/q\mathbb{Z})^*$.

Averaging over the modulus as well, it is possible to obtain fully unconditional estimates valid in a similarly wide range: this is a result due to Barban [2] and Davenport and Halberstam [7]. We will use the following formulation due to Gallagher [11], as stated in [6, Ch. 29].

▶ **Theorem D** (Barban–Davenport–Halberstam). *For any fixed $A > 0$ and any Q such that $x(\log x)^{-A} < Q < x$, we have:*

$$\sum_{q\le Q}\sum_{a\in(\mathbb{Z}/q\mathbb{Z})^*} \left|\pi(x;q,a) - \frac{\pi(x)}{\varphi(q)}\right|^2 \ll_A \frac{xQ}{\log x}.$$

Finally, we will also need a few classical facts regarding Euler's totient function (for example, [14, Th. 328 & 330]).

▶ **Lemma E.** *The following asymptotic estimates hold:*

$$\varphi(q) \gg \frac{q}{\log\log q}, \tag{4}$$

$$\Phi(x) := \sum_{q\le x}\varphi(q) = \frac{3x^2}{\pi^2} + O(x\log x). \tag{5}$$

3 Close-to-uniform prime number generation with fewer random bits

3.1 Basic algorithm

A simple method to construct obviously uniformly distributed prime numbers up to x is to pick random numbers in $\{1,\dots,\lfloor x\rfloor\}$ and retry until a prime is found. However, this method consumes $\log_2 x$ bits of randomness per iteration (not counting the amount of randomness consumed by primality testing), and hence an expected amount of $(\log x)^2/\log 2$ bits of randomness to produce a prime, which is quite large.

As mentioned in the introduction, we propose the following algorithm to generate almost uniform primes while consuming fewer random bits: first fix an integer:

$$q \propto x^{1-\varepsilon} \tag{6}$$

and pick a random $a \in (\mathbb{Z}/q\mathbb{Z})^*$. Then, search for prime numbers $\le x$ of the form $p = a + t\cdot q$. This method, described as Algorithm 1, only consumes $\log_2 t = \varepsilon\log_2 x$ bits of randomness per iteration, and the probability of success at each iteration is $\sim \frac{\pi(x;q,a)}{x/q}$. Assuming that Conjecture B is true, which ensure that (3) holds in the range (6), this probability is about $q/\big(\varphi(q)\log x\big)$, and the algorithm should thus consume roughly:

$$\varepsilon\cdot\frac{\varphi(q)}{q}\cdot\frac{(\log x)^2}{\log 2} \tag{7}$$

Algorithm 1 Our basic algorithm.

1: Fix $q \propto x^{1-\varepsilon}$
2: $a \xleftarrow{\$} (\mathbb{Z}/q\mathbb{Z})^*$ ▷ considered as an element of $\{1, \dots, q-1\}$
3: **repeat** forever
4: $\quad t \xleftarrow{\$} \{0, \dots, \lfloor \frac{x-a}{q} \rfloor\}$
5: $\quad p \leftarrow a + t \cdot q$
6: $\quad$ **if** p is prime **then return** p
7: **end repeat**

bits of randomness on average: much less than the trivial algorihm. Moreover, we can also show, under the same assumption, that the output distribution is statistically close to uniform and has close to maximal entropy.

We establish those results in §3.2, and show in §3.3 that Turán's theorem can be used to obtain nearly the same results under the Extended Riemann Hypothesis. ERH is not sufficient to prove that Algorithm 1 terminates almost surely, or to bound the expectation of the number of random bits it consumes, due to the possibly large contribution of negligibly few values of a. We can avoid these problems by modifying the algorithm slightly, as discussed in §3.4. Finally, in §3.5, we show that unconditional results of the same type can be obtained using the Barban–Davenport–Halberstam theorem, for another slightly different variant of the algorithm.

Before turning to these analyses, let us make a couple of remarks on Algorithm 1. First, note that one is free to choose q in any convenient way in the range (6). For example, one could choose q as the largest power of 2 less than $x^{1-\varepsilon}$, so as to make Step 2 very easy. It is preferable, however, to choose q as a (small multiple of a) primorial, to minimize the ratio $\varphi(q)/q$, making it as small as $\propto 1/\log\log q \sim 1/\log\log x$; this makes the expected number of iterations and the expected amount (7) of consumed randomness substantially smaller. In that case, Step 2 becomes slightly more complicated, but this is of no consequence.

Indeed, our second observation is that Step 2 is always negligible in terms of running time and consumed randomness compared to the primality testing loop that follows. Indeed, even the trivial implementation—namely, pick a random $a \in \{0, \dots, q-1\}$ and try again if $\gcd(a, q) \neq 1$—requires $q/\varphi(q) \ll \log\log q$ iterations on average. It is thus obviously much faster than the primality testing loop, and consumes $\ll \log x \log\log x$ bits of randomness, which is negligible compared to (7). Furthermore, an actual implementation would take advantage of the known factorization of q and use a unit generation algorithm such as the one proposed by Joye and Paillier [15], which we can show requires only $O(1)$ iterations on average.

Finally, while we will not discuss the details of the primality test of Step 6, and shall pretend that it returns exact results, we note that it is fine (and in practice preferable) to use a probabilistic compositeness test such as Miller–Rabin instead, provided that the number of rounds is set sufficiently large as to make the error probability negligible. Indeed, the output distribution of our algorithm then stays statistically close to uniform, and the number of iterations is never larger.

3.2 Analysis under the Friedlander–Granville–Montgomery conjecture

As mentioned above, it is straightforward to deduce from the Friedlander–Granville–Montgomery conjecture that Algorithm 1 terminates almost surely, and to bound its expected number of

iterations and amount of consumed randomness.

▶ **Theorem 3.2.1.** *Assume that Conjecture B holds. Then Algorithm 1 terminates almost surely, requires* $(1+o(1))\varphi(q)/q \cdot \log x$ *iterations of the main loop on average, and consumes:*

$$\big(\varepsilon + o(1)\big) \cdot \frac{\varphi(q)}{q} \cdot \frac{(\log x)^2}{\log 2}$$

bits of randomness on average.

Proof. Indeed, fix $q \propto x^{1-\varepsilon}$. Conjecture B implies, uniformly over $a \in (\mathbb{Z}/q\mathbb{Z})^*$:

$$\left|\pi(x;q,a) - \frac{\pi(x)}{\varphi(q)}\right| \ll (x/q)^{1/2} \cdot x^{\varepsilon/4} \propto x^{3\varepsilon/4},$$

which is negligible compared to $\pi(x)/\varphi(q) \gg x^{\varepsilon}/\log x$. As a result, we get $\pi(x;q,a) = (1+o(1))\pi(x)/\varphi(q) = (1+o(1))/\varphi(q) \cdot x/\log x$ uniformly over a, and the success probability of the main loop becomes:

$$\frac{\pi(x;q,a)}{1+\lfloor\frac{x-a}{q}\rfloor} = \frac{q}{\varphi(q)} \cdot \frac{1+o(1)}{\log x}$$

which implies the stated results immediately. ◀

Now let X be the output distribution of Algorithm 1, i.e. the distribution on the set of prime numbers $\leq x$ such that Algorithm 1 outputs a prime p with probability exactly $\Pr[X = p]$. Clearly, we have, for all $(a,q) = 1$ and all t such that $a + t \cdot q \leq x$ is prime:

$$\Pr[X = a + t \cdot q] = \frac{1}{\varphi(q)} \cdot \frac{1}{\pi(x;q,a)}.$$

As a result, the squared Euclidean imbalance of X is:

$$\begin{aligned}
\Delta_2^2(X) &= \sum_{a\in(\mathbb{Z}/q\mathbb{Z})^*} \sum_{a+tq\leq x \text{ prime}} \left|\Pr[X = a+tq] - \frac{1}{\pi(x)}\right|^2 + \sum_{p|q} \frac{1}{\pi(x)^2} \\
&= \sum_{a\in(\mathbb{Z}/q\mathbb{Z})^*} \pi(x;q,a)\left|\frac{1}{\varphi(q)} \cdot \frac{1}{\pi(x;q,a)} - \frac{1}{\pi(x)}\right|^2 + \sum_{p|q} \frac{1}{\pi(x)^2} \\
&= \frac{1}{\pi(x)^2} \sum_{a\in(\mathbb{Z}/q\mathbb{Z})^*} \frac{1}{\pi(x;q,a)}\left|\pi(x;q,a) - \frac{\pi(x)}{\varphi(q)}\right|^2 + \sum_{p|q} \frac{1}{\pi(x)^2} \\
&\ll \frac{1}{\pi(x)^2} \sum_{a\in(\mathbb{Z}/q\mathbb{Z})^*} \frac{1}{x^{\varepsilon}} \cdot x^{3\varepsilon/2} \ll \frac{\log^2 x}{x^2} \cdot \varphi(q) x^{\varepsilon/2} \ll \frac{\log^2 x}{x^{1+\varepsilon/2}} \ll \frac{1}{x^{1+\varepsilon/3}}.
\end{aligned}$$

We can then deduce the following.

▶ **Theorem 3.2.2.** *Assume that Conjecture B holds. Then the output distribution of Algorithm 1 is statistically close to uniform, and its collision entropy is only negligibly smaller than that of the uniform distribution.*

Proof. Indeed, by (2), the statistical distance to the uniform distribution satisfies:

$$\Delta_1(X) \leq \Delta_2(X)\sqrt{\pi(x)} \ll \frac{1}{x^{1/2+\varepsilon/6}} \sqrt{\frac{x}{\log x}} \ll x^{-\varepsilon/6},$$

which is negligible. Moreover, the collision probability is:

$$\beta(X) = \frac{1}{\pi(x)} + \Delta_2^2(X) = \frac{1}{\pi(x)}\left(1 + O\Big(\frac{\pi(x)}{x^{1+\varepsilon/3}}\Big)\right) = \frac{1}{\pi(x)}\left(1 + o\big(x^{-\varepsilon/3}\big)\right).$$

Hence:

$$H_2(X) = \log_2\big(\pi(x)\big) - \log_2\big(1 + o(x^{-\varepsilon/3})\big) = (H_2)_{\max} - o(x^{-\varepsilon/3})$$

as required. ◀

3.3 Analysis under the Extended Riemann Hypothesis

Assume the Extended Riemann Hypothesis, and denote by α the fraction of all possible choices of $a \in (\mathbb{Z}/q\mathbb{Z})^*$ such that the error term $E(x;q,a) := \big|\pi(x;q,a) - \pi(x)/\varphi(q)\big|$ satisfies $E(x;q,a) > x^{3\varepsilon/4}$. Then, Turán's theorem asserts that:

$$\frac{1}{\varphi(q)} \sum_{a \in (\mathbb{Z}/q\mathbb{Z})^*} E(x;q,a)^2 \ll x(\log x)^2,$$

and the left-hand side is greater or equal to $\alpha x^{3\varepsilon/2}$ by definition of α. As a result, we get:

$$\alpha \ll \frac{(\log x)^2}{x^{\varepsilon/2}}$$

and hence α is negligible. Therefore, for all except at most a negligible fraction of choices of $a \in (\mathbb{Z}/q\mathbb{Z})^*$, we obtain that $E(x;q,a) \leq x^{3\varepsilon/4}$, and since $\pi(x)/\varphi(q) \gg x^{\varepsilon}/\log x$, this implies $\pi(x;q,a) = (1 + o(1))\pi(x)/\varphi(q)$ as before. As a result, under ERH, we obtain an analogue of Theorem 3.2.1 valid with overwhelming probability on the choice of a.

▶ **Theorem 3.3.1.** *Assume ERH holds. Then Algorithm 1 terminates with overwhelming probability. Moreover, except for a negligible fraction of choices of the class $a \bmod q$, it requires $(1 + o(1))\varphi(q)/q \cdot \log x$ iterations of the main loop on average, and consumes:*

$$\big(\varepsilon + o(1)\big) \cdot \frac{\varphi(q)}{q} \cdot \frac{(\log x)^2}{\log 2}$$

bits of randomness on average.

Turning now to the output distribution of the algorithm, we now note that Algorithm 1 almost surely produces an output for a given choice of a if and only if $\pi(x;q,a) \neq 0$, and this is no longer certain under ERH. Therefore, the probability that the algorithm outputs a prime $p = a + tq \leq x$ becomes:

$$\Pr[X = a + tq] = \frac{1}{\varphi_x^*(q)} \cdot \frac{1}{\pi(x;q,a)},$$

where $\varphi_x^*(q) = \#\{a \in (\mathbb{Z}/q\mathbb{Z})^* \mid \pi(x;q,a) \neq 0\}$. By the previous discussion on the distribution of the values $\pi(x;q,a)$, we know that $\varphi_x^*(q) = \varphi(q) \cdot \big(1 - O(\frac{\log^2 x}{x^{\varepsilon/2}})\big)$. As a result, a similar computation as in §3.2 gives:

$$\Delta_2^2(X) = \frac{1}{\pi(x)^2} \sum_{\substack{a \in (\mathbb{Z}/q\mathbb{Z})^* \\ \pi(x;q,a) \neq 0}} \frac{1}{\pi(x;q,a)} \Big|\pi(x;q,a) - \frac{\pi(x)}{\varphi_x^*(q)}\Big|^2 + \frac{\omega(q)}{\pi(x)^2},$$

where $\omega(q)$ denotes as usual the number of prime factors of q. Then, using the coarse lower bound $\pi(x;q,a) \geq 1$ when $\pi(x;q,a) \neq 0$, we get:

$$\Delta_2^2(X) \leq \frac{S^2 + \omega(q)}{\pi(x)^2}$$

where:

$$\begin{aligned}
S &= \sqrt{\sum_{a\in(\mathbb{Z}/q\mathbb{Z})^*} \left|\pi(x;q,a) - \frac{\pi(x)}{\varphi_x^*(q)}\right|^2} \\
&\leq \sqrt{\sum_{a\in(\mathbb{Z}/q\mathbb{Z})^*} \left|\pi(x;q,a) - \frac{\pi(x)}{\varphi(q)}\right|^2} + \sqrt{\sum_{a\in(\mathbb{Z}/q\mathbb{Z})^*} \pi(x)^2 \left|\frac{\varphi(q) - \varphi_x^*(q)}{\varphi(q)\cdot\varphi_x^*(q)}\right|^2} \\
&= O(x^{1/2}\log x) + \sqrt{\varphi(q)\pi(x)^2 \left|\frac{\alpha}{1-\alpha}\cdot\frac{1}{\varphi(q)}\right|^2} \\
&\ll x^{1/2}\log x + \frac{\pi(x)\log^2 x}{\varphi(q)^{1/2}x^{\varepsilon/2}} \ll x^{1/2}\log x + \frac{x\log x\log\log q}{q^{1/2}x^{\varepsilon/2}} \ll x^{1/2}\log x\log\log x.
\end{aligned}$$

by Turán's theorem again. Hence:

$$\Delta_2^2(X) \ll \frac{x\log^{2+\varepsilon} x}{\pi(x)^2} \ll \frac{\log^{3+\varepsilon} x}{\pi(x)}.$$

This is enough to obtain a relatively good bound on the collision entropy:

$$H_2(X) = \log_2(\pi(x)) - \log_2(\log^{3+\varepsilon} x) = (H_2)_{\max} - O(\log\log x)$$

but isn't sufficient for bounding the statistical distance. However, a direct computation using Turán's theorem and the Cauchy–Schwarz inequality is enough:

$$\begin{aligned}
\Delta_1(X) &= \sum_{\substack{a\in(\mathbb{Z}/q\mathbb{Z})^* \\ \pi(x;q,a)\neq 0}} \pi(x;q,a)\left|\frac{1}{\varphi_x^*(q)}\cdot\frac{1}{\pi(x;q,a)} - \frac{1}{\pi(x)}\right| + \sum_{p\mid q}\frac{1}{\pi(x)} \\
&= \frac{1}{\pi(x)}\sum_{\substack{a\in(\mathbb{Z}/q\mathbb{Z})^* \\ \pi(x;q,a)\neq 0}} \left|\pi(x;q,a) - \frac{\pi(x)}{\varphi_x^*(q)}\right| + \frac{\omega(q)}{\pi(x)} \\
&\ll \frac{1}{\pi(x)}\cdot S\cdot\sqrt{\varphi_x^*(q)} \ll \frac{\log x}{x}\cdot x^{1/2}\log^2 x\cdot\sqrt{q} \ll \frac{\log^3 x}{x^{1/2}}\cdot x^{1/2-\varepsilon/2} \ll \frac{\log^3 x}{x^{\varepsilon/2}}.
\end{aligned}$$

We thus obtain the following result.

▶ **Theorem 3.3.2.** *Assume ERH holds. Then the output distribution of Algorithm 1 is statistically close to uniform, and its collision entropy is only $O(\log\log x)$ bits smaller than that of the uniform distribution.*

3.4 Achieving almost sure termination under ERH

Theorem 3.3.1 above is somewhat unsatisfactory, as we have to ignore a negligible but possibly nonzero fraction of all values $a \bmod q$ to obtain a bound on the average number of iterations and on the randomness consumed by Algorithm 1 under ERH. But this is unavoidable for that algorithm: as mentioned above, it is not known whether ERH implies

that for $q \propto x^{1-\varepsilon}$, all $a \in (\mathbb{Z}/q\mathbb{Z})^*$ satisfy $\pi(x;q,a) \neq 0$. And if an a exists such that $\pi(x;q,a) = 0$, the choice of that a in Step 2 of Algorithm 1, however unlikely, is a case of non-termination: as a result, the existence of such an a prevents any nontrivial bound on average running time or average randomness.

We propose to circumvent that problem by falling back to the trivial algorithm (pick a random $p < x$, check whether it is prime and try again if not) in case too many iterations of the main loop have been carried out. This variant is presented as Algorithm 2.

Algorithm 2 A variant which terminates almost surely under ERH.

Fix $q \propto x^{1-\varepsilon}$
$a \xleftarrow{\$} (\mathbb{Z}/q\mathbb{Z})^*$ ▷ considered as an element of $\{1, \dots, q-1\}$
repeat $T = \log^2 x$ times
    $t \xleftarrow{\$} \{0, \dots, \lfloor \frac{x-a}{q} \rfloor\}$
    $p \leftarrow a + t \cdot q$
    if p is prime **then return** p
end repeat
repeat forever
    $p \xleftarrow{\$} \{1, \dots, \lfloor x \rfloor\}$
    if p is prime **then return** p
end repeat

Clearly, since Algorithm 2 is the same as Algorithm 1 except for the possible fallback to the trivial algorithm, which has a perfectly uniform output distribution, the output distribution of the variant is at least as close to uniform as the original algorithm. In other words, the analogue of Theorem 3.3.2 holds, with the same proof.

▶ **Theorem 3.4.1.** *Assume ERH holds. Then the output distribution of Algorithm 2 is statistically close to uniform, and its collision entropy is only $O(\log\log x)$ bits smaller than that of the uniform distribution.*

Moreover, as claimed above, we can obtain the following stronger analogue of Theorem 3.3.1.

▶ **Theorem 3.4.2.** *Assume ERH holds. Then Algorithm 2 terminates almost surely, requires $(1 + o(1))\varphi(q)/q \cdot \log x$ iterations of the main loop on average, and consumes:*

$$\big(\varepsilon + o(1)\big) \cdot \frac{\varphi(q)}{q} \cdot \frac{(\log x)^2}{\log 2}$$

bits of randomness on average.

Proof. Algorithm 2 terminates almost surely because the trivial algorithm does. One can estimate its average number of iterations as follows. Denote by $\varpi(t)$ the probability that Algorithm 2 terminates after exactly t iterations, and $\varpi_a(t)$ the probability of the same event conditionally to a being chosen in Step 2. We have:

$$\varpi_a(t) = \begin{cases} \left(1 - \dfrac{\pi(x;q,a)}{1 + \lfloor \frac{x-a}{q} \rfloor}\right)^{t-1} \cdot \dfrac{\pi(x;q,a)}{1 + \lfloor \frac{x-a}{q} \rfloor} & \text{for } t \leq T; \\ \left(1 - \dfrac{\pi(x;q,a)}{1 + \lfloor \frac{x-a}{q} \rfloor}\right)^{T} \left(1 - \dfrac{\pi(x)}{\lfloor x \rfloor}\right)^{t-T-1} \cdot \dfrac{\pi(x)}{\lfloor x \rfloor} & \text{otherwise.} \end{cases}$$

$$\varpi(t) = \frac{1}{\varphi(q)} \sum_{a \in (\mathbb{Z}/q\mathbb{Z})^*} \varpi_a(t).$$

Moreover, the expected number N of iterations in Algorithm 2 is given by $N = \sum_{t\geq 1} t\varpi(t)$. We can denote by $N_a = \sum_{t\geq 1} t\varpi_a(t)$ the contribution of a certain choice $a \in (\mathbb{Z}/q\mathbb{Z})^*$.

Now, recall from the previous section that $\pi(x; q, a)$ is within a distance $\ll x^{3\varepsilon/4}\log x$ of $E[Y] = \pi^*(x)/\varphi(q)$, except for at most $\varphi(q)\cdot x^{-\varepsilon/2}$ choices of a. If we denote by A the set of "bad" choices of a, we can write, for all $a \in A$:

$$\varpi_a(t) \leq \begin{cases} 1 & \text{for } t \leq T; \\ \Big(1 - \dfrac{\pi(x)}{\lfloor x\rfloor}\Big)^{t-T-1} \cdot \dfrac{\pi(x)}{\lfloor x\rfloor} & \text{otherwise.} \end{cases}$$

Hence, if we let $\xi := \pi(x)/\lfloor x\rfloor$, we get:

$$N_a \leq \sum_{t=1}^{T} t + \sum_{t=T+1}^{+\infty} t(1-\xi)^{t-T-1}\xi = T(T+1)/2 + \sum_{k=1}^{+\infty}(T+k)(1-\xi)^{k-1}\xi$$

$$N_a \leq T(T+1)/2 + T\frac{\xi}{\xi} + \frac{\xi}{\xi^2} \leq T(T+3)/2 + \frac{1}{\xi} \ll \log^4 x.$$

On the other hand, for $a \notin A$, we have $\xi_a := \frac{\pi(x;q,a)}{1+\lfloor\frac{x-a}{q}\rfloor} = \frac{q}{\varphi(q)}\cdot\frac{1+o(1)}{\log x}$. Therefore:

$$N_a = \sum_{t=1}^{T} t(1-\xi_a)^{t-1}\xi_a + (1-\xi_a)^T \sum_{t=T+1}^{+\infty} t(1-\xi)^{t-T-1}\xi$$

$$N_a = \frac{1}{\xi_a} - \sum_{t=T+1}^{+\infty} t(1-\xi_a)^{t-1}\xi_a + (1-\xi_a)^T \sum_{t=T+1}^{+\infty} t(1-\xi)^{t-T-1}\xi$$

$$\Big|N_a - \frac{1}{\xi_a}\Big| \leq (1-\xi_a)^T \sum_{k=1}^{+\infty}\Big[(T+k)(1-\xi)^{k-1}\xi + (T+k)(1-\xi_a)^{k-1}\xi_a\Big]$$

$$\Big|N_a - \frac{1}{\xi_a}\Big| \leq \exp(-T\xi_a)\cdot(2T + 1/\xi + 1/\xi_a)$$

$$\Big|N_a - \frac{1}{\xi_a}\Big| \leq \exp\Big(-(1+o(1))\frac{q}{\varphi(q)}\log x\Big)\cdot(2T + 1/\xi + 1/\xi_a) \ll \frac{1}{x^{1-\varepsilon}}.$$

As a result, we obtain:

$$N = \frac{1}{\varphi(q)}\sum_{a\in(\mathbb{Z}/q\mathbb{Z})^*} N_a = \Big(1 - O(1/x^{\varepsilon/2})\Big)\cdot\Big(\frac{1}{\xi_a} + O(1/x^{1-\varepsilon})\Big) + O(1/x^{\varepsilon/2})\cdot O(\log^4 x)$$

$$= \frac{1}{\xi_a} + O\Big(\frac{\log^4 x}{x^{\varepsilon/2}}\Big) = (1+o(1))\frac{\varphi(q)}{q}\cdot\log x$$

as required. As for the expected number R of random bits consumed by the algorithm, it is given (ignoring the negligible amount necessary to pick a) by:

$$R = \frac{\log x}{\log 2}\Big(\sum_{t=1}^{T}\varepsilon t\cdot\varpi(t) + \sum_{t=T+1}^{+\infty}(\varepsilon T + t - T)\cdot\varpi(t)\Big)$$

and the stated estimate is obtained by an exactly analogous computation. ◀

3.5 An unconditional algorithm

Finally, we propose yet another variant of our algorithm for which both almost sure termination and uniformity bounds can be established unconditionally. It is presented as Algorithm 3.

The idea is to no longer use a fixed modulus q, but to pick it uniformly at random instead in the range $\{1, \ldots, Q\}$ where $Q \propto x(\log x)^{-A}$; uniformity bounds can then be deduced from the Barban–Davenport–Halberstam theorem. Unfortunately, since Q is only polynomially smaller than x, we can no longer prove that the output distribution is statistically close to uniform: the statistical distance is polynomially small instead, with an arbitrarily large exponent depending only on the constant A. On the other hand, termination is obtained as before by falling back to the trivial algorithm after a while, and since q is often very close to x, we get an even better bound on the number of consumed random bits.

Algorithm 3 An unconditional variant.

1: Fix $Q \propto x(\log x)^{-A}$ even
2: $q \xleftarrow{\$} \{Q/2+1, \ldots, Q\}$
3: $a \xleftarrow{\$} \{0, \ldots, q-1\}$
4: **if** $\gcd(a, q) \neq 1$ **then goto** step 2
5: **repeat** $T = \log^2 x$ times
6: $\quad t \xleftarrow{\$} \{0, \ldots, \lfloor \frac{x-a}{q} \rfloor\}$
7: $\quad p \leftarrow a + t \cdot q$
8: $\quad$ **if** p is prime **then return** p
9: **end repeat**
10: **repeat** forever
11: $\quad p \xleftarrow{\$} \{1, \ldots, \lfloor x \rfloor\}$
12: $\quad$ **if** p is prime **then return** p
13: **end repeat**

Algorithm 3 picks the pair (q, a) uniformly at random among pairs of integers such that $q \in \{Q/2+1, \ldots, Q\}$ and a is a standard representative of the classes in $(\mathbb{Z}/q\mathbb{Z})^*$. There are:

$$F(Q) := \sum_{Q/2 < q \leq Q} \varphi(q) = \Phi(Q) - \Phi(Q/2) = \frac{9}{4\pi^2} Q^2 + O(Q \log Q)$$

possible such pairs, and we claim that for all except a polynomially small fraction of them, $\pi(x; q, a)$ is close to $\pi(x)/\varphi(q)$. Indeed, denote by α the fraction of all pairs (q, a) such that:

$$E(x; q, a) := \left| \pi(x; q, a) - \frac{\pi(x)}{\varphi(q)} \right| > (\log x)^{3A/4}.$$

Since $\pi(x)/\varphi(q) \gg x/Q \propto (\log x)^A$, we get $\pi(x; q, a) = (1 + o(1))\pi(x)/\varphi(q)$ for all pairs (q, a) except a fraction of at most α. Moreover, we have the following trivial lower bound:

$$\sum_{Q/2 < q \leq Q} \sum_{a \in (\mathbb{Z}/q\mathbb{Z})^*} E(x; q, a)^2 \geq \left[\alpha F(Q)\right] \cdot (\log x)^{3A/2}.$$

On the other hand, the Barban–Davenport–Halberstam theorem ensures that the sum on the left-hand side is $\ll xQ/\log Q$. As a result, we get:

$$\alpha \ll \frac{(\log x)^{-3A/2}}{F(Q)} \cdot \frac{xQ}{\log Q} \ll \frac{x(\log x)^{-3A/2}}{Q \log Q} \ll \frac{x(\log x)^{-3A/2}}{x(\log x)^{-A+1}} \ll \frac{1}{(\log x)^{A/2}}.$$

This allows us to deduce the analogue of Theorem 3.4.2 for Algorithm 3.

▶ **Theorem 3.5.1.** *Algorithm 3 with $A > 6$ terminates almost surely, requires $(1+o(1))\varphi(q)/q \cdot \log x$ iterations of the main loop on average, and consumes:*

$$\big(A + o(1)\big) \cdot \frac{\varphi(q)}{q} \cdot \frac{\log x \log\log x}{\log 2}$$

bits of randomness on average.

Proof. Algorithm 3 terminates almost surely because the trivial algorithm does. One can estimate its average number of iterations as follows. Denote by $\varpi(t)$ the probability that Algorithm 3 terminates after exactly t iterations, and $\varpi_{q,a}(t)$ the probability of the same event conditionally to the pair (q,a) being chosen in Steps 2–4. We have:

$$\varpi_{q,a}(t) = \begin{cases} \left(1 - \dfrac{\pi(x;q,a)}{1+\lfloor\frac{x-a}{q}\rfloor}\right)^{t-1} \cdot \dfrac{\pi(x;q,a)}{1+\lfloor\frac{x-a}{q}\rfloor} & \text{for } t \le T; \\ \left(1 - \dfrac{\pi(x;q,a)}{1+\lfloor\frac{x-a}{q}\rfloor}\right)^{T} \left(1 - \dfrac{\pi(x)}{\lfloor x\rfloor}\right)^{t-T-1} \cdot \dfrac{\pi(x)}{\lfloor x\rfloor} & \text{otherwise.} \end{cases}$$

$$\varpi(t) = \frac{1}{F(Q)} \sum_{Q/2<q\le Q} \sum_{a\in(\mathbb{Z}/q\mathbb{Z})^*} \varpi_{q,a}(t).$$

Moreover, the expected number N of iterations in Algorithm 2 is given by $N = \sum_{t\ge1} t\varpi(t)$. We can denote by $N_{q,a} = \sum_{t\ge1} t\varpi_{q,a}(t)$ the contribution of a certain choice (q,a).

As we have just seen, $\pi(x;q,a) = (1+o(1))\pi(x)/\varphi(q)$, except perhaps for a fraction $\alpha \ll (\log x)^{-A/2}$ of all choices of (q,a). If we denote by A the set of "bad" choices of (q,a), we can write, as before, that for all $(q,a) \in A$:

$$\varpi_{q,a}(t) \le \begin{cases} 1 & \text{for } t \le T; \\ \left(1 - \dfrac{\pi(x)}{\lfloor x\rfloor}\right)^{t-T-1} \cdot \dfrac{\pi(x)}{\lfloor x\rfloor} & \text{otherwise.} \end{cases}$$

Hence, if we let $\xi = \pi(x)/\lfloor x\rfloor$, we again obtain:

$$N_{q,a} \le \sum_{t=1}^{T} t + \sum_{t=T+1}^{+\infty} t(1-\xi)^{t-T-1}\xi \le T(T+3)/2 + \frac{1}{\xi} \ll \log^4 x.$$

On the other hand, for $(q,a) \notin A$, we have $\xi_{q,a} := \frac{\pi(x;q,a)}{1+\lfloor\frac{x-a}{q}\rfloor} = \frac{q}{\varphi(q)} \cdot \frac{1+o(1)}{\log x}$. Therefore, as before:

$$\Big|N_{q,a} - \frac{1}{\xi_{q,a}}\Big| \le \exp\Big(-(1+o(1))\frac{q}{\varphi(q)}\log x\Big) \cdot (2T + 1/\xi + 1/\xi_{q,a}) \ll \frac{1}{x^{1-\varepsilon}}.$$

As a result, we get:

$$\begin{aligned} N &= \frac{1}{F(Q)} \sum_{Q/2<q\le Q} \sum_{a\in(\mathbb{Z}/q\mathbb{Z})^*} N_{q,a} \\ &= \Big(1 - O\big((\log x)^{-A/2}\big)\Big) \cdot \Big(\frac{1}{\xi_{q,a}} + O(1/x^{1-\varepsilon})\Big) + O\big((\log x)^{-A/2}\big) \cdot O(\log^2 x) \\ &= \frac{1}{\xi_{q,a}} + O\big((\log x)^{4-A/2}\big) = (1+o(1))\frac{\varphi(q)}{q} \cdot \log x \end{aligned}$$

as required, since $4 - A/2 < 1$. As for the expected number R of random bits consumed by the algorithm, it is now given by:

$$R = \frac{A\log\log x}{\log 2} \sum_{t=1}^{T} t \cdot \varpi(t) + \sum_{t=T+1}^{+\infty} \Big(T \cdot \frac{A\log\log x}{\log 2} + (t-T) \cdot \frac{\log x}{\log 2}\Big) \cdot \varpi(t)$$

where we have again ignored the random bits necessary to pick the pair (q, a), since we need only $\frac{Q(3Q+2)}{8F(Q)} \sim \pi^2/6$ iterations of the loop from Step 2 to Step 4 to select it, and hence $O(\log x)$ random bits. The stated estimate is obtained by essentially the same computation as for N. ◀

We now turn to estimates on the uniformity of the output distribution of the algorithm. For that purpose, we consider instead the output distribution X of Algorithm 4, the variant of Algorithm 3 in which no bound is set to the number of iterations of the main loop (Steps 5–9), i.e. with no fallback to the trivial algorithm. Clearly, since the trivial algorithm has a perfectly uniform output distribution, the output distribution of Algorithm 3 is at least as close to uniform as X.

Algorithm 4 An variant of Algorithm 3 with no fallback.

1: Fix $Q \propto x(\log x)^{-A}$ even
2: $q \xleftarrow{\$} \{Q/2+1, \dots, Q\}$
3: $a \xleftarrow{\$} \{0, \dots, q-1\}$
4: **if** $\gcd(a, q) \neq 1$ **then goto** step 2
5: **repeat** forever
6: $\quad t \xleftarrow{\$} \{0, \dots, \lfloor \frac{x-a}{q} \rfloor\}$
7: $\quad p \leftarrow a + t \cdot q$
8: $\quad$ **if** p is prime **then return** p
9: **end repeat**

Now, Algorithm 4 produces an output (almost surely) for exactly those choices of (q, a) such that $\pi(x; q, a) \neq 0$. Let us denote by $F_x^*(Q)$ the number of such choices. Clearly, with notation from above, we have:

$$F_x^*(Q) = \sum_{Q/2<q\leq Q} \varphi_x^*(q) \qquad \text{and} \qquad 1 - \alpha \leq \frac{F_x^*(Q)}{F(Q)} \leq 1.$$

Then, the probability that Algorithm 4 outputs a given prime $p \leq x$ can be written as:

$$\Pr[X = p] = \frac{1}{F_x^*(Q)} \sum_{\substack{Q/2<q\leq Q \\ p\nmid q}} \frac{1}{\pi(x; q, p \bmod q)}.$$

Therefore, we have:

$$\begin{aligned}
\Delta_1(X) &= \sum_{p\leq x} \left| \Pr[X = p] - \frac{1}{\pi(x)} \right| \\
&= \sum_{p\leq x} \left| \frac{1}{F_x^*(Q)} \sum_{\substack{Q/2<q\leq Q \\ p\nmid q}} \frac{1}{\pi(x; q, p \bmod q)} - \frac{1}{\pi(x)} \right| \\
&= \frac{1}{F_x^*(Q)} \sum_{p\leq x} \left| \sum_{\substack{Q/2<q\leq Q \\ p\nmid q}} \frac{1}{\pi(x; q, p \bmod q)} - \sum_{Q/2<q\leq Q} \frac{\varphi_x^*(q)}{\pi(x)} \right| \\
&\leq \frac{1}{F_x^*(Q)} \sum_{p\leq x} \sum_{\substack{Q/2<q\leq Q \\ p\nmid q}} \left| \frac{1}{\pi(x; q, p \bmod q)} - \frac{\varphi_x^*(q)}{\pi(x)} \right| + \frac{1}{F_x^*(Q)} \sum_{p\leq x} \sum_{\substack{Q/2<q\leq Q \\ p\mid q}} \frac{\varphi_x^*(q)}{\pi(x)}
\end{aligned}$$

$$\leq \frac{1}{F_x^*(Q)} \sum_{Q/2<q\leq Q} \sum_{\substack{p\leq x \\ p\nmid q}} \left| \frac{1}{\pi(x;q,p \bmod q)} - \frac{\varphi_x^*(q)}{\pi(x)} \right| + \frac{1}{F_x^*(Q)} \sum_{Q/2<q\leq Q} \frac{\omega(q)\varphi_x^*(q)}{\pi(x)}$$

$$\leq \frac{1}{F_x^*(Q)\pi(x)} \sum_{\substack{Q/2<q\leq Q \\ a\in(\mathbb{Z}/q\mathbb{Z})^*}} \left| \pi(x) - \varphi_x^*(q)\pi(x;q,a) \right| + \frac{O(\log Q)}{\pi(x)}.$$

We can then bound the sum over (q,a) of $\left|\pi(x) - \varphi_x^*(q)\pi(x;q,a)\right|$ as $D + D^*$, where:

$$D = \sum_{\substack{Q/2<q\leq Q \\ a\in(\mathbb{Z}/q\mathbb{Z})^*}} \left| \pi(x) - \varphi(q)\pi(x;q,a) \right| \quad \text{and} \quad D^* = \sum_{\substack{Q/2<q\leq Q \\ a\in(\mathbb{Z}/q\mathbb{Z})^*}} \left| \varphi(q) - \varphi_x^*(q) \right| \cdot \pi(x;q,a).$$

Now, on the one hand:

$$\begin{aligned} D^* &= \sum_{Q/2<q\leq Q} \left| \varphi(q) - \varphi_x^*(q) \right| \sum_{a\in(\mathbb{Z}/q\mathbb{Z})^*} \pi(x;q,a) \\ &\leq \left(F(Q) - F_x^*(Q) \right) \cdot \pi(x) \leq \alpha F(Q)\pi(x) \ll x^3 (\log x)^{-5A/2-1}, \end{aligned}$$

and on the other hand, applying the Cauchy–Schwarz inequality and the Barban–Davenport–Halberstam theorem:

$$\begin{aligned} D &= \sum_{\substack{Q/2<q\leq Q \\ a\in(\mathbb{Z}/q\mathbb{Z})^*}} \varphi(q) \cdot \left| \pi(x;q,a) - \frac{\pi(x)}{\varphi(q)} \right| \leq Q \sum_{\substack{Q/2<q\leq Q \\ a\in(\mathbb{Z}/q\mathbb{Z})^*}} \left| \pi(x;q,a) - \frac{\pi(x)}{\varphi(q)} \right| \\ &\leq Q \cdot \sqrt{F(Q)} \cdot \sqrt{\frac{xQ}{\log Q}} \ll x^3 (\log x)^{-5A/2-1/2}. \end{aligned}$$

As a result, we obtain:

$$\Delta_1(X) \ll \frac{1}{F_x^*(Q)\pi(x)} \cdot (D + D^*) \ll \frac{(\log x)^{2A+1}}{x^3} \cdot \frac{x^3}{(\log x)^{5A/2+1/2}} \ll \frac{1}{(\log x)^{(A-1)/2}}.$$

We can thus state:

▶ **Theorem 3.5.2.** *The output distribution of Algorithm 3 has a statistical distance $\Delta_1 \ll (\log x)^{(1-A)/2}$ to the uniform distribution. In particular, this distance is polynomially small as long as $A > 1$.*

4 Comparison with other prime number generation algorithms

In this section, we compare other prime number generation algorithms to our method. In §4.1 we show that the output distribution Brandt and Damgård's PRIMEINC algorithm exhibits significant biases (a fact that is intuitively clear, but which we make quite precise), and in §4.2, we discuss the advantages and drawbacks of our method compared to that of Maurer, as the only previous work emphasizing a close to uniform output distribution.

4.1 PRIMEINC

Previous works on the generation of prime numbers, such as [3, 15], provide a proof (based on rather strong assumptions) that the output distribution of their algorithm has an entropy not much smaller than the entropy of the uniform distribution. This is a reasonable measure of

the inability of an adversary to guess which particular prime was output by the algorithm, but it doesn't rule out the possibility of gaining some information about the generated primes. In particular, it doesn't rule out the existence of an efficient distinguisher between the output distribution and the uniform one.

Consider for example the PRIMEINC algorithm studied by Brandt and Damgård in [3]. In essence, it consists in picking a random integer $y < x$ and returning the smallest prime greater or equal to y. There are some slight technical differences between this description and the actual PRIMEINC[5], but they have essentially no bearing on the following discussion, so we can safely ignore them.

It is natural to suspect that the distribution of the output of this algorithm is quite different from the uniform distribution: for example, generating the second prime of a twin prime pair, i.e. a prime p such that $p-2$ is also prime, is abnormally unlikely. More precisely, if one believes the twin prime conjecture, the proportion of primes of that form among all primes up to x should be $\sim 2c_2/\log x$, where $c_2 \approx 0.66$ is the twin prime constant. On the other hand, PRIMEINC outputs such a p if and only if it initially picks y as p or $p-1$. Therefore, we expect the frequency of such primes p in the output of PRIMEINC to be much smaller, about $4c_2/(\log x)^2$. This provides an efficient distinguisher between the output of PRIMEINC and the uniform distribution.

More generally, it is easy to see that the method used in [3] to obtain the lower bound on the entropy of the output distribution of PRIMEINC can also provide a relatively large constant lower bound on the statistical distance to the uniform distribution. Indeed, the method relies on the following result, obtained by a technique first proposed by Gallagher [12].

▶ **Lemma F** ([3, Lemma 5]). *Assume the prime r-tuple conjecture, and let $F_h(x)$ denote the number of primes $p \leq x$ such that the largest prime q less than p satisfies $p - q \leq h$. Then for any constant λ,*

$$F_{\lambda \log x}(x) = \frac{x}{\log x}\left(1 - e^{-\lambda}\right)(1 + o(1))$$

as $x \to +\infty$.

Now, the probability that the PRIMEINC algorithm outputs a fixed p can clearly be written as $d(p)/x$, where $d(p)$ is the distance between p and the prime that immediately precedes it. Let Δ_1' be the statistical distance between the output distribution of PRIMEINC and the uniform distribution. We have:

$$\Delta_1' = \sum_{p \leq x} \left| \frac{d(p)}{x} - \frac{1}{\pi(x)} \right| > \sum_{\substack{p \leq x \\ d(p) > 2\log x}} \left| \frac{2\log x}{x} - \frac{\log x}{x} \right|$$

for $x \geq 17$ in view of the already mentioned classical bound $\pi(x) > x/\log x$ for $x \geq 17$. By Lemma F, this gives:

$$\Delta_1' > \frac{\log x}{x} F_{2\log x}(x) = \left(1 - e^{-2}\right)(1 + o(1)) > 0.86 + o(1)$$

as $x \to +\infty$, and in particular, Δ_1' admits a relatively large constant lower bound (at least if one believes the prime r-tuple conjecture on which the entropy bound is based).

Since the entropy lower bounds for other prime generation algorithms like [15] are based on the same techniques, it is very likely that their output distributions can similarly be shown to be quite far from uniform.

---

[5] To wit, Brandt and Damgård restrict their attention to odd numbers $x/2 < y < x$, and set an upper bound to the number of iterations in the algorithm

4.2 Maurer's algorithm

In [18, 19], Maurer proposed a prime generation algorithm based on a similar principle as Bach's technique to produce random numbers with known factorization [1]. This algorithm has the major advantage of producing a primality certificate as part of its output; as a result, it generates only provable primes, contrary to our method.

Additionally, in the standard version of the algorithm, the distribution of generated primes is heuristically close to uniform. More precisely, this distribution would be exactly uniform if the following assertions held:

1. The distribution of the relative sizes of the prime factors of an integer x of given length *conditional to* $2x + 1$ being prime is the same as the distribution of the relative sizes of the prime factors of a uniformly random integer of the same length.
2. That distribution is, moreover, the same as the asymptotic one (i.e. as the length goes to $+\infty$), as computed using Dickson's ρ function.

Assertion 1 is a rather nonstandard statement about prime numbers, but Maurer points to some heuristic arguments that renders it quite plausible [17], at least asymptotically. Assertion 2 is of course not exactly true, and it seems difficult to quantify how much this affects the output distribution, but probably not to a considerable extent.

That standard version of the algorithm, however, has a small probability of not terminating, as discussed in [19, §3.3]. To circumvent that problem, Maurer suggests a modification to the algorithm which introduces biases in the output distribution. If it is used, the algorithm will in particular never generate primes p such that $(p-1)/2$ has a large prime factor. This provides an efficient distinguisher from the uniform distribution (exactly analogous to the "twin prime" distinguisher of the previous section), since e.g. safe primes have non negligible density. There are other ways to avoid cases of non-termination that do not affect the output distribution to such an extent (such as some form of backtracking), but Maurer advises against them because of the large performance penalty they may incur.

Furthermore, the paper [19] also describes faster variants of the algorithm that are less concerned with the quality of the output distribution, and they do indeed have outputs that are very far from uniform, typically reaching only around 10% of all prime numbers.

More importantly, the main drawback of Maurer's algorithm compared to our method is likely the amount of randomness it consumes: it is within a small constant factor of the amount consumed by the trivial algorithm. In fact, if we neglect everything but the last loop of the topmost step in the recursion, and count as zero the cases when the recursion has more than two steps, we see that the average amount of randomness used by the algorithm is bounded below by c times the corresponding amount for the trivial algorithm, where (with the notations of [19, Appendix 1]):

$$c \geq \int_{1/2}^{1} (1-x)\, dF_1(x) = \int_{1/2}^{1} (1-x)\, \frac{dx}{x} = \log 2 - \frac{1}{2} \geq 0.19$$

and we only counted a very small fraction of the actual randomness used!

As a result, in context where random bits are scarce and primality certificates are not required, it seems that our method offers a rather better trade-off than Maurer's algorithm: the uniformity of the distribution is easier to estimate in our case, and we use much less randomness. Moreover, Maurer finds [19, §4.3] that an efficient implementation of his algorithm is about 40% slower than the trivial algorithm, whereas Algorithm 1 is easily 5 times as fast in practice.

On the other hand, if proofs of primality are considered important, it is clearly better to rely on Maurer's algorithm than to use our method and then primality proving.

References

1 E. Bach. How to generate factored random numbers. *SIAM J. Comput.*, 17(2):179–193, 1988.

2 M. B. Barban. The "large sieve" method and its application to number theory. *Uspehi Mat. Nauk*, 21:51–102, 1966.

3 J. Brandt and I. Damgård. On generation of probable primes by incremental search. In E. F. Brickell, editor, *CRYPTO*, volume 740 of *Lecture Notes in Computer Science*, pages 358–370. Springer, 1992.

4 J. Brandt, I. Damgård, and P. Landrock. Speeding up prime number generation. In H. Imai, R. L. Rivest, and T. Matsumoto, editors, *ASIACRYPT*, volume 739 of *Lecture Notes in Computer Science*, pages 440–449. Springer, 1991.

5 R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. *ACM Trans. Inf. Syst. Secur.*, 3(3):161–185, 2000.

6 H. Davenport. *Multiplicative Number Theory*, volume 74 of *Graduate Texts in Mathematics*. Springer, 2nd edition, 1980.

7 H. Davenport and H. Halberstam. Primes in arithmetic progressions. *Michigan Math. J.*, 13:485–489, 1966.

8 C.-J. de la Vallée Poussin. Recherches analytiques sur la théorie des nombres premiers. *Ann. Soc. Sci. Bruxelles*, 20:281–397, 1896.

9 D. Eastlake 3rd, J. Schiller, and S. Crocker. Randomness Requirements for Security. RFC 4086 (Best Current Practice), June 2005.

10 J. B. Friedlander and A. Granville. Limitations to the equi-distribution of primes I. *Ann. Math.*, 129:363–382, 1989.

11 P. X. Gallagher. The large sieve. *Mathematika*, 14(1):14–20, 1967.

12 P. X. Gallagher. On the distribution of primes in short intervals. 23:4–9, 1976.

13 G. H. Hardy and J. E. Littlewood. Some problems of 'partitio numerorum': III. on the expression of a number as a sum of primes. 44:1–70, 1922.

14 G. H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers*. Clarendon Press, 4th edition, 1960.

15 M. Joye and P. Paillier. Fast generation of prime numbers on portable devices: An update. In L. Goubin and M. Matsui, editors, *CHES*, volume 4249 of *Lecture Notes in Computer Science*, pages 160–173. Springer, 2006.

16 M. Joye, P. Paillier, and S. Vaudenay. Efficient generation of prime numbers. In Çetin Kaya Koç and C. Paar, editors, *CHES*, volume 1965 of *Lecture Notes in Computer Science*, pages 340–354. Springer, 2000.

17 U. Maurer. Some number-theoretic conjectures and their relation to the generation of cryptographic primes. In C. Mitchell, editor, *Cryptography and Coding '92*, pages 173–191. Oxford University Press, Mar. 1992.

18 U. M. Maurer. Fast generation of secure RSA-moduli with almost maximal diversity. In *EUROCRYPT*, pages 636–647, 1989.

19 U. M. Maurer. Fast generation of prime numbers and secure public-key cryptographic parameters. *J. Cryptology*, 8(3):123–155, 1995.

20 P. Mihăilescu. Fast generation of provable primes using search in arithmetic progressions. In Y. Desmedt, editor, *CRYPTO*, volume 839 of *Lecture Notes in Computer Science*, pages 282–293. Springer, 1994.

21 P. Mihăilescu. Security of biased sources for cryptographic keys. In *Cryptography and computational number theory (Singapore, 1999)*, volume 20 of *Progr. Comput. Sci. Appl. Logic*, pages 287–302. Birkhäuser, Basel, 2001.

22 H. L. Montgomery. *Topics in Multiplicative Number Theory*, volume 227 of *Lecture Notes in Mathematics*. Springer, 1971.

23 H. L. Montgomery. Problems concerning prime numbers. *Proc. Symp. Pure Math.*, 28:307–310, 1976.

24 V. Shoup. *A Computational Introduction to Number Theory and Algebra (Version 2)*. Cambridge University Press, 2008.

25 P. Turán. Über die Primzahlen der arithmetischen Progression. *Acta Sci. Math. Szeged*, 8(4):226–235, 1936.

26 A. Walfisz. Zur additiven Zahlentheorie II. *Mathematische Zeitschrift*, 40(1):592–607, 1936.